MapReduce for distributed Data Mining

Thomas Jacobs
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email: tjacobs@mail.upb.de

Abstract

Having a huge collection of data, a typical task is to analyze this data regarding certain objectives. Often, such analyzing algorithms can be decomposed into special map and reduce sub-tasks. This paradigm is well known as MapReduce, originally published by Google. A key benefit of this method is that it is straight forward to distribute the sub-tasks to multiple machines making use of parallelism. This way, analyzing huge data collections becomes feasible. However, the original architecture of MapReduce was rather intended to be used on homogenous high performance clusters. This paper investigates the challenges to bring MapReduce to peer-to-peer networks consisting of heterogeneous machines with different capacities and where nodes frequently join and leave. Existing literature on this topic is presented and evaluated. Based on these investigations, possible integrations and uses in the Peer-to-Peer Framework in general and LifeSocial.KOM in special are discussed.

1 Introduction

We tend to describe ourselves as living in an information society. Gathering new knowledge is crucial and information is a central economic good. However, as Han et al. pointed out, we live in a world, which is *data rich but information poor* [16, p.5]. Data is collected everywhere we leave our digital footprints, resulting in massive data collections. Taking a look at Facebook for example, half a petabyte of new data arrives at their warehouse every 24 hours [8]. But there is an important difference between collecting data and having knowledge. To gather new knowledge, the data repositories have to be processed and analyzed. Meaningful data patterns must be extracted and evaluated. This process is well known as **data mining** [16, p. 8]. Now, having these on growing collections of data, the question is, how these datasets can be processed in a feasible and yet efficient way. An intimating idea is to make use of data parallelism. That is, distribute the data to several machines and process these data blocks in parallel. This is the central idea, the MapReduce paradigm makes use of [11]. A key benefit is that programmers using MapReduce do not have to care about work distribution and parallel computation. All this is automatically handled by the MapReduce framework in the background. Basically, the only thing which has to be done is to decompose the algorithm to process the data into special map and reduce functions. This might be a main reason, why MapReduce, since published in 2004 by Google, became a great success story. A lot of literature has been published, describing ways to make well known algorithms work in the MapReduce environment [6]. Furthermore, multiple open source implementations of MapReduce like Skynet [7] and Apaches Hadoop [2] have emerged.

However, as originally proposed by Google, MapReduce was rather designed to be implemented in environments consisting of homogenous, high-performance clusters. This paper now focuses on the challenges to bring MapReduce to peer-to-peer networks. The basic idea is to use the computation power of existing machines, instead of having to pay for a dedicated server-cluster. New machines can dynamically enter and leave the network offering their capabilities to be used. Node failures are likely to happen and machines joining the network have different capacities. Furthermore, nodes can be malicious and cannot be trusted automatically. This paper presents and discusses existing literature on adapting MapReduce to peer-to-peer environments. Based on these investigations, possible adaptions to and uses in the Peer-To-Peer Framework and LifeSocial.KOM by the University of Paderborn are discussed [14]. The usage of MapReduce in this context seems to be quite promising, since data in LifeSocial.KOM is already spread over and fragmented at several nodes throughout the network. Considering bandwidth being a bottleneck, processing the data locally at the nodes in parallel instead of transferring everything to a requestor most likely results in a computation-speedup.

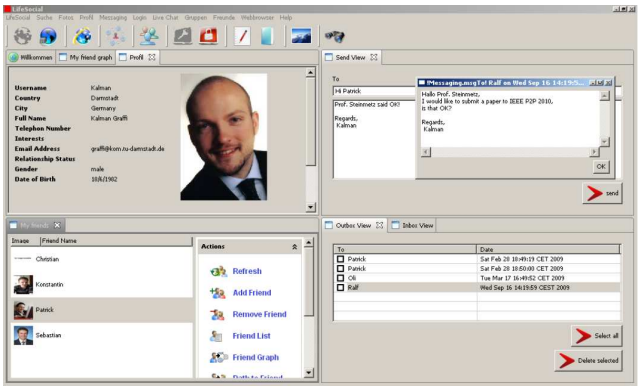


Figure 1: Screenshot of LifeSocial.KOM (source: [14])

The remainder of this paper is structured as follows: Section 2 briefly introduces the Peer-to-Peer Framework. Section 3 describes the general idea of MapReduce and how Google has integrated it into their cluster environment. Section 4 then investigates the challenges to bring MapReduce to peer-to-peer environments. Several approaches found in literature are discussed. Finally, in Section 5, possible strategies to bring MapReduce to the Peer-to-Peer Framework and LifeSocial.KOM are presented and possible applications of MapReduce are pointed out.

2 Overview of the Peer-to-Peer Framework and LifeSocial.KOM

The Peer-to-Peer Framework developed by the University of Paderborn is a decentralized network entirely based on Pastry [14][1]. Pastry is a structured overlay network acting on top of the public internet. It can be seen as an abstraction bringing along a new addressing and storage scheme. Machines (also referred to as peers or nodes) are identified by unique ids and hash tables distributed at the peers are used for routing messages. Delivery of messages, provided the target node exists and the network is stabilized, is guaranteed after passing $O(log(N))$ hops, whereas N is the number of peers. There are no central servers, but instead all data is distributed throughout the network. Since peers dynamically leave and enter the network, PAST [4] is used and has been extended to provide proper data replication. Furthermore, principles of symmetric and asymmetric cryptography are used to provide access control, confidentiality and encryption [15].
To show the practical feasibility of using the Peer-To-Peer Framework, LifeSocial.KOM has been developed on top of it [14]. It is a social network providing a set of plugins, which offer user profiles, a chat system, group communication, friend lists and many more features [13]. Figure 1 shows a screenshot of LifeSocial.KOM. By using the notion of plugins and since plugins can use each other, it is highly adaptable and extendable.

3 Fundamentals of MapReduce

Before discussing possible adaptions to peer-to-peer networks and LifeSocial.KOM in special, it is worthwhile to look at the basic principles of MapReduce and how it is used in Google's environment.

3.1 Basic principles

Taking the view of a software developer, his task is to express the data processing algorithm in terms of so-called **map** and **reduce** functions. The signatures of these functions are defined as follows [11]:

$$\begin{aligned} map &: Key \times Val \implies list(Key2 \times Val2) \\ reduce &: Key2 \times list(Val2) \implies list(Val2) \end{aligned}$$

The map function takes a key and a value as input. The value is then processed and a list of new keys and new values, possibly from another domain, is emitted. Before executing the reduce operation, all values val2 corresponding to the same key2 are summarized in a common list. Then, the reduce function processes each key2 with the corresponding list of values val2 to generate the final output list for that key.

user	date	time (min)
ameier	09-08-12	154
fhuber	09-08-12	20
ameier	11-08-12	200
kmax	13-08-12	20

Table 1: Example log-file

This description seems to be quite abstract at the first glance. Let us therefore consider a small example. Suppose having a set of log-files. Let a line of such a log-file at least consist of a username and the amount of time this user has worked with a system (Table 1). Using MapReduce, the aim is to output for each user the total time he spent working at some system. Algorithm 1 lists the corresponding map and reduce functions to obtain the results. The map function takes a document as input and emits for each line the username and the time the user has spent on the system. This results in a list of tuples, i.e. $(ameier, 154), (fhuber, 20), (ameier, 200), ...$ Before executing the reduce function, values (i.e. times) corresponding to one key (i.e. username) are mapped to a list. For $ameier$, as an example, this results in list $(154, 200)$. Finally, for each user the reduce function is executed, summing-up the times. As a result, for each user the total time is emitted.

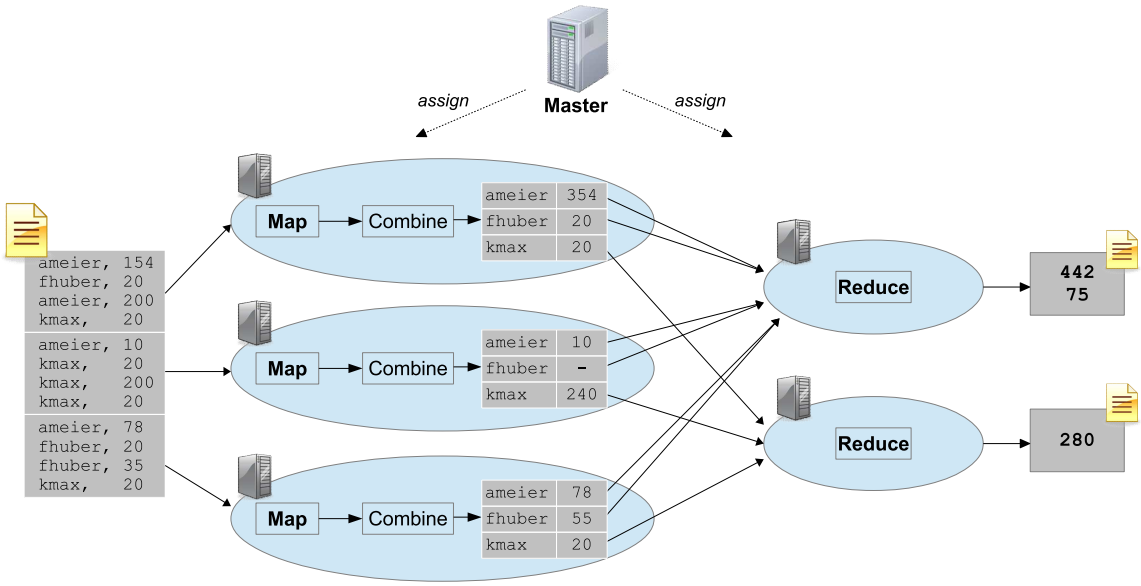


Figure 2: Executing a MapReduce job

Algorithm 1 Computing elapsed user times

```
function MAP(String key, String value)
    //key = name of log-file
    //value = contents of log-file
    for each line li in value do
        Emit(username,time)
    end for
end function
function REDUCE(String key, Iterator value)
    //key = username
    //values = list of time-values
    int time = 0
    for each v in value do
        time+ = v
    end for
    Emit(toString(time))
end function
```

3.2 Distributed execution

As we have seen from the previous sub-section, the software developer did not have to worry about work distribution and parallel computation. This is a key advantage of MapReduce. We can look at MapReduce as a high-level framework. We tell MapReduce what jobs are to be executed on which input data and MapReduce will automatically take care of job scheduling and task distribution. This sub-section now takes a look behind the scenes, how MapReduce works in Google's computing environment.

In Google's environment, MapReduce is executed on clusters, in which files are managed by the Google File System (GFS) [11]. Making use of MapReduce, the aim is to process and analyze these files in parallel. Figure 2 presents an overview, how a MapReduce job is executed in this environment for our example Algorithm 1. The central component is a master workstation, which is part of the cluster. The MapReduce job is handed over to this master and it is then responsible to schedule and assign corresponding map and reduce tasks to workers, also considered as slaves. In the following, a slave executing a map-task (reduce-task) is called a map-worker (reduce-worker). However, note that any slave is able to execute map and reduce tasks, possibly both at the same time. As a first step, the input data is split into logical blocks. The master assigns these blocks to map-workers. Note that the workers executing map tasks can run in parallel, each one working on an own data block. Going back to our example from above (Algorithm 1), for each username a list of time-values is generated as intermediate result, i.e. $(ameier, (154, 200))$. These intermediate results are stored on the local discs of the map-workers. However, a reduce task can be executed on a different machine and the intermediate data has to be transferred to that machine. Therefore, it is useful to let the user define a so-called combine task. This task usually does the same as the reduce function, but it is executed on the map-worker on his intermediate results. It aggregates the values for each key and thereby reduces the data size. In our example it would transform $(ameier, (154, 200))$ to $(ameier, (354))$, decreasing the amount of data to be transferred to a reduce-worker. As a result, each map-worker produces intermediate data

of $key, value$ pairs. When a map-worker has finished his task, it informs the master about the locations of intermediate data residing on his local disc. As a next step, the master assigns reduce tasks to slaves. It partitions the intermediate keys into blocks and assigns each block of keys with the corresponding intermediate values to a reduce worker. This partition has to be defined by the user, since each partition will result in a separate output-file. In Figure 2 for example, the first reduce-worker processes data for keys $ameier$ and $fhuber$, whereas the second reduce-worker processes data of $kmax$. Each reduce-worker transfers the data to be processed into his local memory and sorts it by keys (usernames). For each key, the reduce function is executed on the corresponding data values and the results are appended to an output file, one for each partition of intermediate keys. These output-files are written to a network location defined by the master. Note that in general, Google assumes that the output-files do not have to be merged together as one. It is guaranteed that all values for one output-key are stored in exactly one output-file. For our example this is easy to observe, since for each username only one output-value, the system time, is computed.

3.3 Assumptions and implications

A Google cluster may consist of thousands of machines [11]. These machines do not necessarily have the same specifications, but are of comparable capability. Network bandwidth is considered to be valuable. Therefore, the master tries to assign map-jobs to machines, on which GFS locally stores the data blocks to be processed. When a slave fails during computation, this is detected by the master and the task is forwarded to another slave. Replication of files is guaranteed by the underlying GFS. As Google only considers a single master handling all MapReduce tasks, a master failure results in all currently running MapReduce tasks to fail. The user has to request the re-execution in this case. Another problem are stragglers. These are machines which need considerably more time to complete a task than the other machines. Since these stragglers can extend the execution time of the whole MapReduce job, the master detects those stragglers and assigns the corresponding map or reduce tasks to be additionally executed by other machines. In this case, only the result of the first ready machine executing the same task is accepted as result. Assuming that MapReduce tasks are deterministic, Google guarantees that the result computed, when there were failures during computation, is the same as the result obrained from a non-faulting sequential execution [11].

4 MapReduce in peer-to-peer environments

After explaining the usage of MapReduce in Google's computing environment, the question now is, which challenges arise when porting this framework to peer-to-peer networks. This section deals with that question and discusses different approaches found in literature.

4.1 Challenges and requirements

As a general starting point, it is worthwhile to look at the well-known 8 fallacies of distributed computing [19]:

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

From this list, we can derive important requirements for the use of MapReduce in peer-to-peer environments in general and LifeSocial.KOM developed by the University of Paderborn in special. First, LifeSocial.KOM is deployed in a public network infrastructure. Thus, the network is not reliable and machines and users may be malicious. Nodes dynamically leave and enter the network and failures are likely to happen. Bandwidth should be considered as valuable and tasks should be executed on nodes locally storing the data to be processed. Since machines are heterogeneous, the different capabilities must be taken into account. There even may be machines not having the capability to execute map or reduce tasks at all. Especially there might arise a problem of the master becoming a bottleneck. Since we suppose to have a decentralized environment, there is no central server which can act as a master. Having mobile devices, network transport costs and battery usage have to be taken into account. These give us the following list of requirements for a solution:

1. Works in conditions with frequent churn
2. Copes with node failures, esp. with master failures
3. Avoids bottleneck machines
4. Makes use of data locality
5. Provides security in an untrustworthy environment
6. Takes heterogenous machines into account
7. Takes mobile nodes into account

Having this list of requirements and comparing it with the assumptions made by Google, the existing solution is not completely sufficient for peer-to-peer environments. The following sub-sections present several proposals found in literature that at least address some of the requirements for peer-to-peer networks.

4.2 Dynamic Master/Slave partitioning

To address the problem of frequent churn and master failures, Marozzo et al. propose a solution, which dynamically partitions the entire peer-to-peer network into

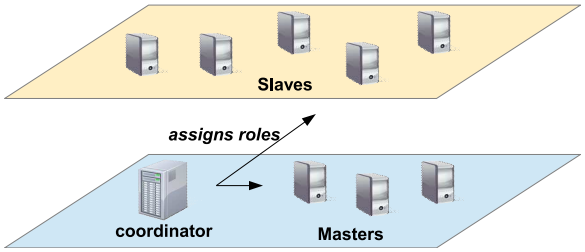


Figure 3: Architecture proposed by Marozzo et al.

master and slave machines [18]. This is, at each time, a certain percentage of the machines are masters. There is a special primary master, referred to as coordinator, which is capable of changing masters into slaves and vice versa. This way, a dynamic adaption is possible. Whenever the coordinator fails, a new one is elected. Figure 3 sketches this architecture. MapReduce tasks now are executed as follows: The user gets a list of masters, chooses the one having the lowest workload and hands over the MapReduce job. This master becomes the primary master for that MapReduce job and it elects two backup masters, which on failure can overtake the role of the primary master. The primary master assigns map and reduce tasks to slaves and keeps track of slave failures.

The proposed solution by Marozzo et al. has been implemented on top of the JXTA peer-to-peer framework [3]. The peers participating in MapReduce computation are organized as a group. Each of these peers publishes advertisements, for example describing his current role (master or slave) and the current workload index. These advertisements can be discovered by other peers asking special rendezvous nodes, which act as routers. Based on the published advertisements, requesting peers can ask for a list of idle masters and slaves.

Critical investigation

Going back to our list of requirements, the proposed solution considers points 1 (frequent churn) and 2 (node failures). Since masters and slaves are chosen based on their current workload, points 3 (bottleneck machines) and 6 (heterogeneous machines) are considered. Furthermore, by additional advertisements, mobile nodes (point 7) can be taken into account. For example, a mobile node could publish advertisements indicating that it is not willing to transfer huge amounts of data, due to transport costs. Currently not considered are points 4 (data locality) and 5 (security). Regarding point 4, a master might prefer slave nodes on which the data to be processed currently is stored onto. But, since each node might become a master, the data to be processed might currently be stored at a master-node and thus has to be transferred to a slave. However, assuming that the fraction of master nodes is

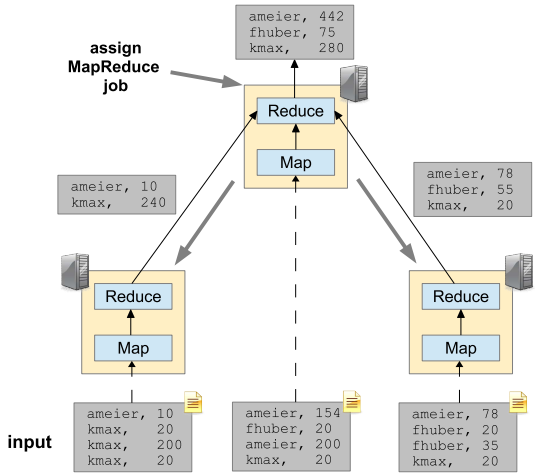


Figure 4: MapReduce using a tree architecture.

small compared to the set of slave nodes, the impacts of this issue are minor. Furthermore note that currently all nodes are treated trustworthy in this solution.

Besides solving most of the requirements, a possible disadvantage can result from the quite involved and complex protocol. A lot of management data has to be exchanged between peers, since the network is actively stabilized and a lot of advertisements are published. In an extended paper [12], Marozzo et al. compared their solution to a centralized MapReduce implementation. In their evaluations, they also took a close look on the amount of network data transferred during their simulation. Although the number of messages sent in the peer-to-peer scenario were considerably higher than in the centralized case, the total amount of data transferred was comparable [12].

4.3 Tree-structured architecture

Another approach to make use of MapReduce in peer-to-peer networks was proposed by Lee et al. [17]. They especially addressed the problem of having a single master node. This node may become a bottleneck and limits scalability. To overcome these problems and to support the decentralized structure of peer-to-peer networks, Lee at al. propose a tree-based architecture to execute MapReduce jobs. In their solution, there does not exist a single master machine, anymore. Instead, each node acts as both, master and slave. All peers participating in MapReduce computations organize themselves in a tree-structure. This tree is established on a continuous sub-region of the id-space of the peer-to-peer network using the distributed hash tables (DHTs). Thereby, Lee et al. guarantee that the latency to propagate a message down the tree is in $O(log(N))$ hops, whereas N is the total number of nodes of the tree. Machines, which enter or leave the network, are added to or pruned from this tree.

Once the tree-structure has been set up, MapReduce jobs

can be executed. This is sketched in Figure 4 for Algorithm 1 and works as follows: First of all, the MapReduce job is sent to the root node of the tree. Each tree node partitions the input data to be processed into blocks. On one of these blocks, a map task is locally executed by that node. The other data blocks are distributed to child nodes which recursively process that blocks in the same way. After executing a map task, the tree node waits for the children to finish their execution. Thereafter, the node executes a reduce operation on the intermediate result of his map task and on the results produced by the reduce tasks of the children. This way, data is aggregated, before forwarding it to the next parent node. Finally, the root node produces the requested result of the MapReduce job.

Critical investigation

A first important thing to note is that the application of the reduce function in this tree-based approach differs in the way it is used in Google's approach. As seen in Figure 2, in Google's solution, a reduce operation is applied to all values for a specific key. As an implication, all map tasks have to be finished when executing reduce tasks. This is different in the tree-based solution. A reduce operation is applied over and over on intermediate data. Using the terminology of Google, the reduce operation is used as a combiner, here. As an advantage, the amount of intermediate data which is forwarded to parent-nodes is kept low. However, Lee et al. implicitly assume the reduce function to be commutative and associative. That is, they assume the order in which reduce operations are applied to the data values for one key does not matter. Although this works for the reduce function of example Algorithm 1 (this is because the addition operator is associative and commutative), this might not always be the case.

Comparing the tree-based approach to the list of requirements, it is best suited for full filling point 3 (avoiding bottlenecks), since there is no more central master. Node failures and churn (points 1 and 2) are only considered to a certain degree. The tree is able to manage itself, pruning failing nodes and allowing new ones to enter. However, there might be the case, in which a node fails during a MapReduce computation. Currently Lee et al. do not have an efficient strategy to provide fault tolerance for this case [17]. Failing nodes are detected by timeouts and their data has to be processed by another node. A major disadvantage of this strategy is that if one slave fails, all data processed by this failing machine and its child nodes has to be processed again. This is especially problematic, when a node close to the root-node of the tree fails. Therefore, more elaborate strategies to provide fault tolerance should be evaluated. Point 4 (data locality) is only considered for intermediate results, which are first aggregated and then sent to parent nodes. This way, the amount of intermediate data transferred through the network is kept low. However, data locality for the input documents to be processed currently is not considered. All input datasets have to be transferred to the corresponding tree nodes in order to execute map-tasks. Currently, point 5 (security) is not considered. Point 6 (heterogeneous nodes) is implicitly ensured, since each node in the tree can decide how much input data it should process by its own. Similarly, incapable mobile nodes can be supported or simply be excluded from the tree.

4.4 Single distributed master architecture

At a first glance, a single master architecture seems not to fullfill our requirements. However, Dang et al. propose an interesting approach which is able to avoid bottlenecks and deals with heterogeneous capabilities of nodes [10]. Basically, Dang et al. exploit two things. First, they implement their solution on top of the Gnutella protocol. Thereby, they use the fact that Gnutella divides peers into normal peers and so-called super peers. By using only the super peers for MapReduce computations, only capable machines take part. Second, each super peer can become a master. There is no single master for all the MapReduce jobs, but instead the (super-) peer requesting to perform a MapReduce job becomes the master for its and only its job. Executing a MapReduce job now works as follows: The requesting peer (i.e. the master) asks neighboring super peers in the network to join his MapReduce group, which is identified by a unique id. Based on the current workload and other factors, a contacted peer can either decide to join or not to join the group. Furthermore, each peer contacted, recursively forwards the join request to his neighbors. A peer joining the group automatically takes the role as a slave node in this group. However, note that a peer can be part of different groups at the same time, provided it has sufficient available resources. As in the proposed solution by Google, the master then assigns map and reduce task to slaves and manages the distributed computation.

Critical investigation

Dang et al. implicitly assume in their solution that only super peers request to perform MapReduce jobs. Using this assumption, a major benefit is that the solution does not have to bother with master failures, since the requesting machine is the master itself. When this machine fails, the result of the MapReduce operation is no more needed and slave nodes eventually time out. This way, point 2 of the requirements (coping with node failures) is satisfied. The same holds for point 1 (churn), since this is managed by the underlying peer-to-peer framework and since MapReduce groups are built upon request only. By letting each peer act as a master for his own MapReduce job and by considering super peers only, bottlenecks can be avoided as long as the master machine is able to manage its workload. A peer can

	Dyn. Master/Slave partitioning *(Marozzo et al.)*	Tree-structured architecture *(Lee et al.)*	Single dist. master architecture *(Dang et al.)*
1. frequent churn	handled	handled	handled
2. node failures	backup masters, coordinator	problematic during MapReduce operations, when machines close to the root node fail	handled
3. bottleneck nodes	nodes chosen based on workload - extendable	no central master	node requesting a MapReduce operation becomes the master, only super peers participate
4. data locality	not considered	not considered	not considered
5. security	not considered	not considered	not considered
6. heterogeneous nodes	nodes chosen based on workload - extendable	tree-nodes can decide how much data to process, incapable nodes can be pruned	nodes can decide to enter a MapReduce group, only super peers considered
7. mobile nodes	*extendable to mobile nodes*	*extendable to mobile nodes*	*extendable to mobile nodes*
further properties	induces a high amount of management and discovery messages	assumes the reduce function to be commutative and associative	high latency for setting-up a new MapReduce group

Figure 5: Overview of different strategies for MapReduce in peer-to-peer environments.

furthermore reject to enter a group, whenever it is busy. As a disadvantage, finding slave nodes for a group might take some time. For reasons already explained, points 6 (heterogeneous machines) and 7 (mobile nodes) can be taken into account, as well. Currently not considered are points 4 (data locality) and 5 (security). However, it seems to be very well possible to find solutions for these problems, as it will be discussed in Section 5.

4.5 Comparison of the approaches

Figure 5 gives on overview of the three different approaches presented in the previous sub-sections. None of the approaches fulfills the whole list of requirements. Furthermore, basically two different architectures were proposed: A tree-based architecture and two master/slave architectures. Comparing the two master/slave architectures the implementation and design decisions are rather orthogonal. Whereas in the solution of Marozzo et al., a lot of effort is spent to cope with node failures and to publish and discover advertisements, the opposite is the case in the solution of Dang et al.. The advantage of the first solution is that a master can pick slaves based on their current workload and a MapReduce job can immediately be executed. In the solution of Dang et al., the node requesting to execute the MapReduce job becomes a master and first has to create a MapReduce group. It does not have information of workloads on other peers and peers can decide not to join his group. Nevertheless, the approach of Dang et al. seems to be quite promising for its simplicity. Messages regarding MapReduce are only exchanged on demand and the management overhead is kept to a very low degree.

5 Developing a solution for the Peer-to-Peer Framework and LifeSocial.KOM

After discussing the different approaches to bring MapReduce to peer-to-peer networks in general, this Section now focuses on developing a possible solution for the Peer-to-Peer Framework and LifeSocial.KOM based on Pastry.

5.1 Possible applications of MapReduce

In general, we can differentiate between two kinds of MapReduce applications. As a first application, we could use the Peer-to-Peer Framework as a tool to process generic datasets. This is, the data to be processed is external to the network and the Peer-to-Peer Framework is used as a tool to analyze it. However, this implies that the datasets have to be transferred to the peers executing map and reduce operations. Because of this, the datasets should be provided by an external sever or peer providing enough bandwidth. Furthermore, the download-speed of the peers participating in the distributed computation should be taken into account. Yet another possibility is to use MapReduce for datasets which entirely are part of the peer-to-peer network. Considering LifeSocial.KOM, we could use MapReduce to analyze user profiles or friend lists for example. Furthermore, users have the possibility to upload documents and we could use MapReduce to execute a distributed keyword search on a set of documents. Since typically, the results generated by reduce operations are aggregated, the resulting output size is much smaller than the input size of the data. Instead of transferring all the input data to a single machine and to process it sequentially, it seems to be more efficient to process the datasets in parallel directly on the peers they are stored on. As another benefit, bandwidth can be preserved this way. The following sub-sections will rather focus on apply-

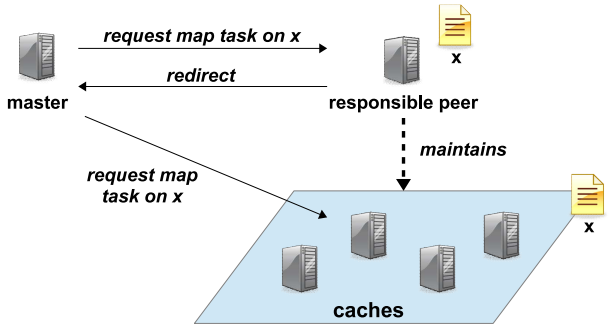


Figure 6: Assigning map tasks using caching.

ing MapReduce for datasets being part of the network, since we can look at it as a special case for the first approach.

5.2 Architectural suggestions

As a first design decision, the architecture has to be chosen. The single distributed master architecture by Dang et al. as described in Section 4.4 seems to be very promising. Their approach is quite simple and the amount of network management data sent around is kept to a low degree. As the node requesting to execute a MapReduce job becomes the master for his job, master failures do not have to be treated separately. However, a peer requesting for a MapReduce job could not have the capacities to work as a master. To solve this problem, we could combine the approach of Dang et al. with the tree-based solution by Lee et al.. Whenever a master peer does not have enough capacity available, it could divide the input data and delegate the work to some other peers which then act as child-masters. This way, we get a tree of depth two. As a disadvantage however, we have to keep in mind that whenever a child-master fails, all the data assigned to that master has to be processed again by a new child-master. As an open topic, more efficient strategies to provide fault-tolerance in this case should be evaluated.

5.3 Assigning tasks to slaves

A master has to elect slave nodes to execute map and reduce operations. Considering the list of requirements, bandwidth is valuable. For this reason, first of all, map tasks should be executed on peers which are responsible for the input data blocks. These peers can be found using the distributed hash table (DHT). However, such a peer might not have enough capacity to execute a map task, or the item requested might be a hotspot. Taking a closer look at the current implementation of the Peer-to-Peer Framework, a responsible node for a data item keeps a list of nodes, which previously requested an item and still have it in cache [5]. Based on a configurable metric, either the responsible node provides the item requested, or the address of a node having that item in cache is returned. This strategy can be easily extended to work with MapReduce. As sketched in Figure 6, the responsible node can redirect the request to process item x to another peer. As another benefit, the data item can be divided into further blocks. The responsible node could hand back a list of caching nodes to the master, which could assign map tasks to several caching nodes, each one processing a different data block of the item. An open question however is, to which peers the responsible node should redirect requests. Basically, two orthogonal solutions are possible, here. One strategy is to select nodes for redirection at random. The other solution is that the responsible node actively keeps track of the current workloads and parameters of the nodes having an item in their cache. This second strategy has the advantage that an actively maintained metric can directly deal with the requirement to cope with heterogeneous nodes. However, a lot of maintenance messages have to be exchanged. Which solution is best suited should be evaluated empirically. Another problem might be the use of erasure codes to provide redundancy. In this scenario, files are divided into a number of k blocks which are then encoded as n symbols. These n symbols are stored distributed throughout the network. The question now is, whether there exist possibilities to process the data without transferring the symbols to one peer and to decode the original file again. Currently, erasure codes are not used in the Peer-to-Peer Framework, but they might be used in the future [5].

Yet another question is, after executing map tasks, how reduce tasks should be distributed. Since intermediate data produced by the map tasks is stored locally at the map-workers, the master should reuse these peers to execute the reduce operations. Nevertheless, as one reduce operation processes all data-values for one intermediate key (compare to Section 3.2) at least some intermediate data has to be shifted to reduce workers. Therefore, combine functions should be executed directly after executing map operations in order to aggregate intermediate results before sending it to reduce-workers. Whenever the amount of data generated by the combine operations is small, the master could also decide to execute a single reduce operation by his own.

5.4 Security considerations

None of the approaches discussed in Section 4 took security issues into account. However, this topic is crucial for LifeSocial.KOM, since it is used on top of the public internet. LifeSocial.KOM already provides a lot of security mechanisms like secure communication, file encryption and access control. These mechanisms have to be extended for MapReduce. Since map and reduce tasks are executed on different machines, these machines could be malicious and produce wrong results. There also can be malicious nodes acting as masters requesting new MapReduce jobs over and over again and thereby overload the network. But most importantly, to execute map and reduce operations, the slaves

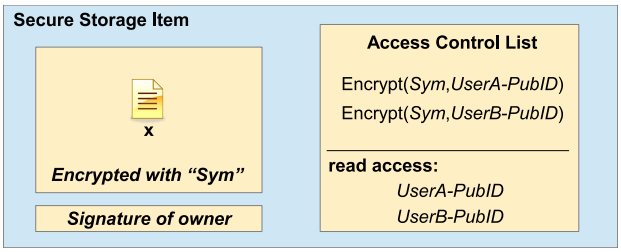


Figure 7: Encryption and access control for data items (extends the solution of [9]).

have to decrypt the encrypted files, which leads to possible security problems.

Finding solutions for the first two problems is beyond the scope of this paper, but these issues have to be properly addressed in future work. Every possible solution to the latter problem must obey the principle that only peers (users) having the right to read a data item are able to decrypt it. This is already guaranteed by the use of access control lists. As sketched in Figure 7, each data item is attached with an access control list resulting in a so-called secure storage item [9]. The data item is encrypted using a (random) symmetric key. For each user having access to that item, the symmetric key is encrypted using the public key of that user and the result is stored to the access control list. Using his private key, the user then can first decrypt the symmetric key and then decrypt the data item. Furthermore, the secure storage item is signed by the owner.

Now, the master has to know which slaves have appropriate access rights to process the input data. This can be supported in an efficient way, if the responsible peer for a secure storage item would know which of the caches have read access to the item. To support this, the access control list in Figure 7 is extended by a list of all public keys of users having read access to the data item. For Figure 6, this would allow the responsible peer to filter out caches not having read access. Further strategies have to be developed for the case, when no caches having read access are available.

After executing the map and combine tasks on the set of slaves, the master has to assign reduce tasks to slaves. Since intermediate data has to be shifted to peers executing reduce operations, these data items should be encrypted again. Only those slaves having the right to access the original data item should have access to the corresponding intermediate data. Therefore, the intermediate data produced by a map/combine operation should be encrypted again with the same symmetric key as for the original data item processed by that map operation. A slave executing a reduce operation then has to decrypt the intermediate data again. This has to be considered for the master when scheduling reduce taks to slaves. There also might arise situations, in which there are no slaves having read access to all intermediate data and the master has to execute a final reduce operation by itself.

5.5 Concluding overview

Looking back at the list of requirements, the proposed solution works for point 1 (frequent churn), since this is handled by the Peer-to-Peer Framework. Whenever a slave peer fails, the master assigns the task to another slave. Assuming the peer requesting the MapReduce operation has enough available capacities, it acts as the master for its own job. Thereby master failures do not have to be treated separately. Otherwise, child-masters are elected, which are replaced on failures. Bottleneck machines (point 3) can be avoided by using idle machines still having a data item in cache. The same way, data locality (point 4) is exploited. Possible strategies to provide security (point 5) were presented. However, scenarios with adversaries attacking the network still have to be analyzed. Furthermore, when there are only few nodes having read access to the data items to be processed, applying MapReduce might be inefficient. As a responsible node for a data item can redirect requests for map tasks to caching peers, heterogeneous nodes can be supported (point 6). Based on the capacities of these caching peers, the responsible node can decide to which ones it should redirect map tasks. An open question still is, how mobile nodes can be supported best. Especially, the situation might arise, in which a mobile node is responsible for a data item to be processed, but without having a list of caches for that item.

6 Summary and conclusion

This paper investigated approaches to bring MapReduce to peer-to-peer environments in general and the Peer-to-Peer Framework and LifeSocial.KOM in special. As data items are stored distributed throughout the entire peer-to-peer network, using MapReduce to process data at the peers they are stored onto seems to be a promising strategy. Three different approaches found in recent literature were discussed. As orthogonal design decisions, these strategies either rely on master/slave or tree-based architectures. None of the proposed solutions fulfills the complete list of requirements needed for applying MapReduce in the Peer-To-Peer Framework in general and LifeSocial.KOM in special. Therefore, based on existing strategies, a new possible approach has been sketched for LifeSocial.KOM. This approach relies on a single distributed master architecture. That is, each node requesting a MapReduce job is the master for its and only its job. For distributing map and reduce tasks to slaves, data locality is considered to preserve bandwidth. Possible security issues were discussed and still open questions were identified. Empirical evaluations still have to show, how good the sketched solution works out in a prototype. Future work has to address the open questions in order to finally come up with a fully working and yet efficient strategy.

References

[1] FreePastry. Web Page. http://www.freepastry.org. (last checked 27.04.2013).

[2] Hadoop. Web Page. http://hadoop.apache.org. (last checked 27.04.2013).

[3] JXTA Web Page. https://jxta.kenai.com. (last checked 23.05.2013).

[4] PAST. Web Page. http://www.freepastry.org/PAST/default.htm. (last checked 27.04.2013).

[5] *Project Group 2012 (Documentation) - A Peer-to-Peer Framework for Social Networks*.

[6] Researches on MapReduce. Web Page. https://sites.google.com/a/fiu.edu/research-on-mapreduce. (last checked 27.04.2013).

[7] Skynet. Web Page. http://skynet.rubyforge.org. (last checked 27.04.2013).

[8] Under the Hood: Scheduling MapReduce jobs more efficiently with Corona. https://www.facebook.com/notes/facebook-engineering/under-the-hood-scheduling-mapreduce-jobs-more-efficiently-with-corona/10151142560538920. (last checked 27.04.2013).

[9] *Project Group 2011 (Documentation) - A Distributed Framework for Social Networks*, 2011.

[10] H. Dang, H. Tran, P. Vu, and A. Nguyen. Applying MapReduce Framework to Peer-to-Peer Computing Applications. In *ICCCI '12: Computational Collective Intelligence. Technologies and Applications*. 2012.

[11] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI '04: Proceedings of the Symposium on Operating Systems Design and Implementation*, 2004.

[12] P. T. Fabrizio Marozzoa, Domenico Taliaa. P2P-MapReduce: Parallel data processing in dynamic Cloud environments. *Journal of Computer and System Sciences*, 78, 2012.

[13] K. Graffi, C. Groß, P. Mukherjee, A. Kovacevic, and R. Steinmetz. LifeSocial.KOM: A P2P-based Platform for Secure Online Social Networks. In *IEEE P2P '10: Proceedings of the International Conference on Peer-to-Peer Computing*, 2010.

[14] K. Graffi, C. Gross, D. Stingl, D. Hartung, A. Kovacevic, and R. Steinmetz. LifeSocial.KOM: A Secure and P2P-based Solution for Online Social Networks. In *IEEE CCNC '11: Proceedings of the International Conference on Consumer Communications and Networking Conference*, 2011.

[15] K. Graffi, P. Mukherjee, B. Menges, D. Hartung, A. Kovacevic, and R. Steinmetz. Practical Security in P2P-based Social Networks. In *IEEE LCN '09: Proceedings of the International Conference on Local Computer Networks*, 2009.

[16] J. Han, M. Kamber, and J. Pei. *Data Mining*. 3rd edition, 2011.

[17] K. Lee, T. W. Choi, A. Ganguly, D. Wolinsky, P. Boykin, and R. Figueiredo. Parallel Processing Framework on a P2P System Using Map and Reduce Primitives. In *IEEE IPDPSW '11: International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, 2011.

[18] F. Marozzo, D. Talia, and P. Trunfio. A Peer-to-Peer Framework for Supporting MapReduce Applications in Dynamic Cloud Environments. In *Cloud Computing: Principles, Systems and Applications*. 2010.

[19] A. Rotem-Gal-Oz. Fallacies of Distributed Computing Explained. http://www.rgoarchitects.com/Files/fallacies.pdf. (last checked 27.04.2013).

Audio and video capturing, streaming and presenting in Java

Christopher Holomek
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email: holomekc@mail.upb.de

Abstract

This paper provides an overview of audio and video capturing, streaming and presenting technologies in Java. Furthermore, methods for collaborative work on textual documents, calendars and draw boards are presented. The focus will be on the history of these technologies and how suitable they are for the Peer-to-Peer Framework.

1 Introduction

Over time, the ways of communication have changed. A supporting factor for this is the progress of technology. One of the first communication medium was the letter, which allows us to write a message to our communication partner. The fact, that the letter needs to be transported physically, implies that this kind of communication can take some time. After that, a new communication medium was introduced: the telephone. From now on, people were able to communicate directly with the help of their voice. The precondition of beeing at home or at a telephone cell changed with the development of the mobile telephone, which made it possible to communicate nearly from everywhere. Recently, a lot of communication is done via computers. Most of these computers are linked together via the internet. This allows us to communicate to everyone who is connected to that global network. At first, communication was only possible from home, because computers were large and it was not comfortable to carry them around. But that changed with the launch of mobile computers, which we also know as notebooks. The recent development of smartphones and tablet pcs give us an even handier version of computers. That means, that it is possible to connect to the internet from nearly everywhere.

To communicate with the help of computers, basically two different kinds of communication exists: The synchronous and asynchronous communication. Asynchronous communication means, that a participant is not waiting for a direct answer. The sending and receiving of messages is asynchronous. The most famous example for this kind of communication is the E-mail. As the name already suggests, it is an electronic letter. On the other hand synchronous communication expects an immediate answer from the communication partner. Examples for synchronous communication are Instant Messaging and Voice over IP (VoIP). With the help of a video camera it is additionally possible to transmit a live video of ourselves.

This paper will focus on the previously mentioned topic. It will give an overview of technologies, which are able to capture, stream and present audio and video data. Because the Peer-to-Peer Framework is written in Java, all mentioned technologies are usable in Java. In the first section technologies are introduced, which allow the use of multimedia directly in Java. The second section focuses on HTML5 and especially on the Web Real-Time Communication (WebRTC). To apply that in Java, access to a webbrowser is required, which can handle HTML5 and WebRTC. Therefore, this chapter will also consider technologies to achieve that. It is discussed, whether the technologies are suitable for the P2P Framework, by considering the platform-independence, the streaming capability and which other features can be given. Chapter 4 presents an overview of technologies, which focus on collaborative work. These technologies allow the users to work together on written documents, calendars and draw boards. The last chapter attempts an overall summary and concludes the paper.

2 Audio and Video integrated in Java

The following chapter describes technologies, which can connect to a video camera and microphone. Moreover, they provide a way to access the audio and video data directly in Java, so that the video can be displayed on a Java component.

2.1 The JMF Framework

The first technology to handle multimedia data in Java is the Java Media Framework (JMF), which was developed and released by Sun, SGI and Intel in 1998 [6, p. 9]. It is a ”[...] free, platform-independent and object oriented media

processing framework" [6, p. 8]. Thus, it can be fully integrated into Java. The JMF provides features like playing, processing and streaming of media data. The last update for JMF was in 2004 and has not been developed further [32, p. 554]. The download page of the JMF offers four different versions [23]. One Cross-platform Java version and three Performance Packs for Linux, Solaris and Windows. A Performance Pack for OSX does not exist. Playing video files on a Mac with the cross-platform version works, but it is very slow and produces a lot of CPU load. This and the fact that it was not possible to get access to the video camera during the test, on a Macbook Pro, makes JMF not suitable for the P2P Framework.
An alternative to JMF is the Freedom for Media in Java (FMJ) framework. Its application programming interface (API) is compatible with JMF [20]. To capture images from a video camera it uses the Larson Technologies Inc. Capturing Images and Video In a Library (LTI-CIVIL) framework. During the test, it was possible to get access to the video camera, but the framerate was limited to approx 8 frames per second (fps). Which equates to the results of LTI-CIVIL on their homepage [14]. The last update of both frameworks FMJ and LTI-CIVIL was in 2007, so updates seem to be unlikely. According to this circumstances, it is not suitable for live video communication.

2.2 OpenIMAJ

The Open Intelligent Multimedia Analysis toolkit for Java (OpenIMAJ) was developed at the University of Southampton. The development started in 2005 until the first open source release was presented in 2011 and is updated until today [27]. OpenIMAJ is written in pure java, but for the video capturing and hardware access, it also uses some native code. It supports Linux, Windows and OSX and runs on a 32 or 64 bit Java Virtual Machine (JVM) [10, p. 691]. Considering mobile devices it supports at least Android. OpenIMAJ combines a "[...] set of libraries and tools for multimedia content analysis and content generation" [27]. It is really simple to use because of the fact, that it is sufficient to embed the set of jars OpenIMAJ uses. But that does not mean, that all provided jars have to be embedded. OpenIMAJ is separated into multiple modules [10, p. 692]. The relevant modules for the P2P Framework are *video-capture* and *audio*. With the *video-capture* module it is possible to get access to the connected cameras. It uses a lightweight native interface for any supported platform [10, p. 692]. To get an image from a camera it is necessary to select one of the devices the native interface provides. It is also possible to change the used resolution and framerate. After that, it is possible to invoke two methods. The method *nextFrame()* indicates the native code to load the next image of the video camera and the method *getImage()* returns that image data as a byte array. This array stores the RGB values for each pixel of the image. With this data it is easy to create an image so that Java can paint it on a *Canvas* for example. During the test on a Macbook Pro it was possible to paint 30 fps with a resolution of 640x480. This is a really good value and more than sufficient for a video chat, as movements are really smooth. To record audio from the microphone the *audio* module is used. This module works with the Java Sound API, which provides audio samples in form of byte arrays. The ambient noise in the test was quite high, but due to the fact, that it was only a notebook microphone it was quite good. In any case the voice was good to understand. Now it can be assumed that it is easy to transfer audio and video data, because of their byte array representation. But with a resolution of 640x480 the created video data has a size of $640 * 480 * 3 = 900$ kB per frame. Three bytes for each RGB value of a pixel. With a frame rate of 25 it is necessary to transmit $25 * 900$ kB $= 22$ MB per second. That is not feasible with a typical internet connection. So a technique is required to compress that data. Java provides some compression techniques for images. So it should be possible to reduce the data to an acceptable size. The same problem exists in case of the audio data. With a 44.1 KHz sampling rate, 16 bits depth and 2 channels the data size is about 44100 Hz $* 16$ bit $* 2$ channels $= 172$ kB per second. This is to big as well and needs to be reduced further. One possible solution is to reduce the quality of the recording. Compressing each frame separately and reducing the audio quality are inefficient approaches to compress multimedia contents. There are many different codecs for that kind of content, which are more appropriate. To get access to these codecs OpenIMAJ uses the Xuggler library. Xuggler describes itself as an "[...] easy way to uncompress, modify, and re-compress any media file (or stream) from Java" [3]. To achieve that Xuggler uses the FFmpeg[1] library. During tests the encoding worked quite good. One second of that encoded audio and video data takes about 160 kB. That is too big for internet streaming, but the quality can be adjusted to an acceptable value. To play the recorded video OpenIMAJ uses Xuggler as well. Additionally, Xuggler enables the possibility to store the video chat, which is a nice to have feature. The disadvantage of Xuggler is, that its size is big. It needs more than 40 MB disk space. OpenIMAJ itself needs about 20-50 MB disk space depending on how many modules are used. Compared to other video chat clients that is a lot of storage capacity. For example, Skype only need approx 18 - 50 MB depending on which platform it is used. But even the *video-capture* module contains more features than needed for the P2P Framework. The only feature needed is the access to the video camera. So it is possible to reduce the disk space of OpenIMAJ, by separating some Java classes and native code from OpenIMAJ modules, which removes all unnecessary dependencies. For au-

[1] FFmpeg is a library for streaming, converting and recording multimedia data which is available for most platforms

dio, OpenIMAJ uses the Java Sound API and this is already part of the Java JRE, so this does not need any disk space. After that OpenIMAJ needs only approx 2 MB plus 40MB for the Xuggler library. A disadvantage of OpenIMAJ is, that it produces higher CPU load. The reason for this is the slow drawing on a Java component.

As already mentioned OpenIMAJ provides many different modules. Therefore, more features besides capturing video and recording audio are available. For example, the *faces* module allows to recognize faces on a video camera picture. It draws rectangles at the positions where faces are recognized. That enables a variety of additional features. Moreover, the *audio* module provides more features besides playing and recording of audio data. For example, it is possible to analyse the waveform of the audio input of the microphone. This can be used to provide a function like voice activation. Therefore, audio is only recorded when the amplitude is above a specified value.

2.3 OpenCV

Open Source Computer Vision (OpenCV) is an open source library with a set of algorithms that provides several functions [13]. The word "Computer Vision" means that the "[...] data from a still or video camera [is transformed] into either a decision or a new representation" [2, p. 2]. For the P2P Framework the input data is a video camera and the decision is "a person who wants to communicate". A new representation might be the horizontal flipping of the video images such that it looks like a person looks into a mirror. That makes it easier for the user to position himself or to hold an object infront of the video camera. OpenCV "[...] is written in C and C++ and runs under Linux, Windows and Mac OS X" [2, p. 1]. The newest version of OpenCV also supports iOS and Android [13]. The development of OpenCV started in 1999 and the first offical release version was published in 2006. It is updated till today. The next feature, which is currently in development is a CUDA[2] interface to speed up some algorithms [12]. Before OpenCV can be used it is necessary to install it separately. For the use of OpenCV in Java it is possible to use JavaCV which provides an interface for OpenCV [21]. To get access to the video camera it is sufficient to select a device via a Java Method. Then it is possible to change some settings, like the resolution, framerate, etc. After that, the invocation of the *grab()* method returns an image of the video camera. This image can be painted on a *Canvas*. With a resolution of 640x480 OpenCV reaches approx 30 fps on a Macbook Pro. This is as fast as OpenIMAJ. The audio data from a microphone can be accessed through the same object which is responsible for the video data. But instead of calling *grab()* the invocation of *grabFrame()* provides access to the audio samples in form of a byte array. The size of the audio and video data is comparable to the values calculated during the test with OpenIMAJ. This means, that with OpenCV it is necessary to compress the data for streaming as well. To store the recorded audio and video data to a hard disk JavaCV uses FFmpeg similar to what Xuggler does. Similar to OpenIMAJ, OpenCV requires more CPU resources for the drawing process.

As mentioned before, OpenCV needs to be installed separatly. Unlike OpenIMAJ, OpenCV is not devided into modules. Thus, it is only possible to install the whole software or nothing. The Windows version of OpenCV is a about 280 MB big .exe file, which can be easily installed. The Linux/OSX version is a lot more difficult to install. It is available as source code and it is necessary to compile and install it before it can be used. To install it on an OSX system, it is necessary to install XCode[3] which is about 1,5 GB big. The next step is to start XCode and download the Command Line Tools provided in the software. After that, another software like MacPorts[4] can use XCode to compile the source of OpenCV. To do this, several commands have to be typed into the OSX Terminal. This is not user friendly and it can be assumed that most users are not willing to use the software when it is already so complex to install it. A possible solution to avoid that is to precompile OpenCV for Linux and OSX or to prepare a batch file for automated installation.

2.4 JavaFX

Java Applets made Java so famous as it is today. But today these applets do not play an important role anymore. Therfore, Oracle tries to provide an alternative approach to Adobe's *Flash* and Microsoft's *Silverlight* by introducing JavaFX in 2008 [33, p. 69]. Originally, it was a script based language. But this changed pretty fast. Already the second version of JavaFX was a pure Java library similar to Swing and SWT [33, p. 69]. To use JavaFX it was necessary to install it separately, but this changed. JavaFX is part of Java since version 7u6 but it is necessary to add JavaFX to the build path before it can be used. For the release of Java 8, which is considered to be in fall of 2013, it is planned to integrate JavaFX into the Java Standard Edition (Java SE) [24]. As mentioned before, JavaFX was planed to be an alternative to Flash and Silverlight. So it is incomprehensible for which reason JavaFX does not provide access to the video camera and microphone, although the competitors do. It is not clear whether the next version of JavaFX will close that gap. Therefore, JavaFX is not an alternative for the P2P Framework in its current version. The biggest competitor for all named technologies in this section is HTML5

[2] CUDA is created by Nvidia and provide access to the grafic processing units (GPUs)

[3] XCode is a development environment for OSX

[4] MacPorts can be used to compile and install software by using a command line tool under OSX

combined with JavaScript. Even Microsoft and Adobe prefer HTML5 already [33, p. 69]. More information about HTML5 will follow in the next chapter. So the future of JavaFX in web applications is uncertain. It is more probable that JavaFX will replace Swing and AWT in client side applications [33, p. 70].

3 Audio and Video in HTML5

In this chapter the paper describes technologies, which are able to render HTML5 pages. Moreover, it is necessary that theses technologies support the WebRTC standard, which provides access to a video camera and microphone. The following section will give an overview, how WebRTC works and how it can be used.

3.1 WebRTC

The beginning of WebRTC was in 2008 when Google's Gmail video chat became more famous. After that Google released Hangouts in 2011, which works on the same technique as the Gmail video chat. To push the development Google bought the company GIPS, which focused on components for Real-Time Communication (RTC) systems. RTC systems "enable human communication via voice and video" like VoIP, Chat and Video Telephony [5]. These components are now released under the name WebRTC, as an open standard, which is currently under development by Google, Mozilla and Opera. WebRTC is also approved by the World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF). The aim of this standard is to provide RTC for web browsers. To achieve that WebRTC uses HTML5 and JavaScript [8]. HTML5 is currently still under development and will be the newest version of Hypertext Markup Language (HTML). HTML allows to add and structure text, images and hypertext directly into a web page. As it is an Extensible Markup Language (XML) based language, it uses tags to structure the content. In the newest version of HTML5 it is possible to add audio and video content to web pages. Adding a video to an HTML document is really simple. The only thing needed is the <*video*> tag and the <*source*> tag. The <*video*> tag positions the video frame on the web page and the <*source*> tag describe, which video content should be played. The second technique WebRTC is using is JavaScript. JavaScript is a script language, which is directly written in an HTML document. It is executed on the client side and can be used to manipulate content dynamically [25, p. 103 ff.].

But how does WebRTC combine these both techniques? The following description is based on a tutorial from Sam Dutton [5]. WebRTC uses two APIs. The first one is *MediaStream*. The most important method of this API is the *getUserMedia(video: true, audio: true, handleStream, handleError)*[5] method. It provides access to the video camera and/or to the microphone. To change the resolution a framerate optional constraints can be used. To display the local video, the stream of the video camera needs to be added as a source of the <*video*> tag. This is done in the *handleStream(stream)* function. The second API is *RTCPeerConnection*. This API is used to stream the data between the peers. But before the data can be transmitted the clients have to be introduced to each other. To achieve that a web server application is required. More concrete, this server handles the so called signaling part of WebRTC. Signaling contains three types of information that need to be exchanged:

1. **Session control messages:** initialize or close communication and report errors.
2. **Network configuration:** What's my computer's IP address and port?
3. **Media capabilities:** What codecs and resolutions do the clients support?

RTCPeerConnection does not specify how this messages are exchanged. Therefore, it is possible to use any appropriate messaging protocol. Possible ones are Session Initiation Protocol (SIP), Extensible Messaging and Presence Protocol (XMPP), etc. For the exchange of media configuration information, WebRTC uses the Session Description Protocol (SDP). This protocol defines a format for video and audio information, such as supported resolutions and codecs. To agree on a codec and resolution an offer and answer message are exchanged. The exchange of network configurations is more complex. When both clients are directly connected to the internet, they can be identified by their ip address. But when client A is behind a router, which is using Network Address Translation (NAT), then the client B cannot access his conversational partner. The problem is that B can access the router, but then the router does not know for which device the data is intended, so it will drop that data. Therefore, WebRTC uses the Interactive Connectivity Establishment (ICE) protocol to solve this problem. During the first step the ICE protocol tries to establish a direct connection between the peers. If that is not possible it uses a Session Traversal Utilities for NAT (STUN) server. Simply spoken, STUN works as follows. The client sends a message to a STUN server. Therefore, the client hand that message out to the NAT. The NAT translates the local address of the client to a public address and forwards the message to the STUN server. The STUN server receives this message and recognizes from where it came from. That means, that the server is now aware of the client's public address. The STUN server sends back that address to the NAT, which now knows where to forward that message. After that, the client can use its public address to communicate with his conversational partner, by sending this information through the web server. When the router uses an

[5] webkit users invoke webkitGetUserMedia(...), for firefox mozGetUserMedia(...)

enterprise NAT and/or a more severe firewall, then STUN is not enough, because a direct connection is not possible. Therefore, the ICE protocol uses a Traversal Using Relays around NAT (TURN) relay server. This means, that both clients communicate with the help of that server. The finding of a communication partner is illustrated in Figure 1. When a direct connection is established, the audio and video

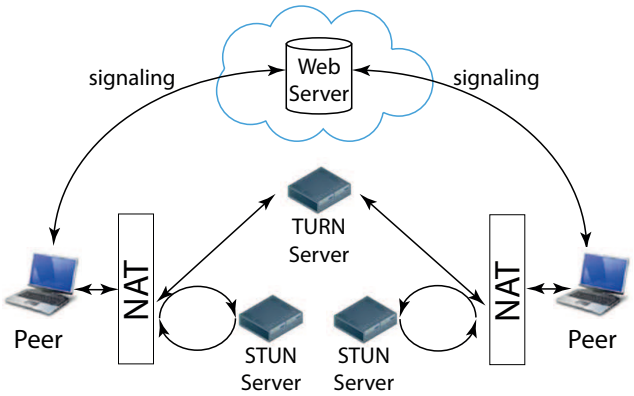


Figure 1: Finding a communication partner
Source: Own figure based on [5]

data can be transmitted directly peer-to-peer. WebRTC uses the Datagram Transport Layer Security (DTLS) protocol for a secure connection. But when this was not possible, the multimedia data has to be transmitted through a relay server. Figure 2 illustrates how the data is transmitted, when a direct connection is established or when a TURN server is needed. Additonally to audio and video data, WebRTC

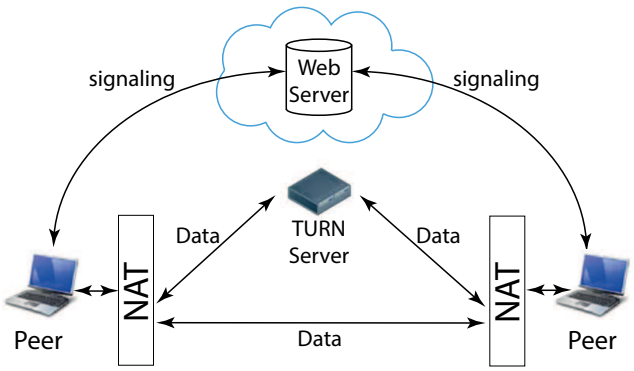


Figure 2: Data transfer in WebRTC
Source: Own figure based on [5]

also allows the transmission of other contents. The RTC-DataChanel API enables features like file transfer, real-time text chat, gaming and much more. But this are more potential features, because the development is still in progress [5].

To make WebRTC usable for the Peer-to-Peer Framework it is necessary to remove the centralized parts of that approach. One possible solution would be to integrate the web server application into the P2P Framework, so that every client has its own server which delegates the communication between the framework and the HTML5 and JavaScript part of WebRTC. Figure 3 illustrates how the

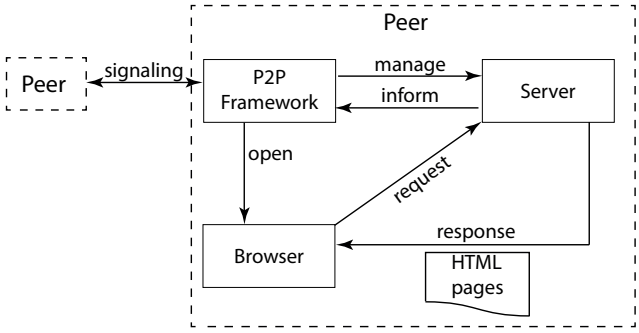


Figure 3: WebRTC and the P2P Framework
Source: Own figure

communication could work. When a user calls another participant, the P2P Framework starts a server process on each client's machine. After that, a browser is opened, which sends a request to this server. The server responds with an HTML document. This document contains the HTML5 and JavaScript part of WebRTC. Before a peer-to-peer connection can be established, the server needs to inform the P2P Framework about all necessary information. The framework now uses its own connection to delegate the messages. When all needed information are exchanged, WebRTC is able to establish a connection and start the video communication. As described, the P2P Framework now handles the signaling part of WebRTC. Therefore, it not necessary to use a server. But there is still a centralized part. The *RTCPeerConnection* API needs a STUN/TURN server, otherwise it is not usable. Therefore it would be possible to use a STUN server of Google, Mozilla, etc. Another possibility would be to implement a STUN server into the P2P Framework, similar to the previous approach for the web server application. A conceivable task for the project group would be to determine, if there is a workaround to use WebRTC without a STUN server. The next subsections will introduce technologies to use HTML5 and WebRTC in combination with Java.

3.2 HTML generation in Java

As explained in chapter 3.1 each client of the Peer-to-Peer Framework will have its own server process. This server will be integrated into the P2P Framework and has to generate the HTML pages. This chapter will introduce a suitable technology, which can achieve that. A possible idea for generating HTML pages is to implement this page directly into a server method. So when the server receives a

request it simply calls this method and returns that HTML page. The disadvantage of that approach is, that it is more difficult to overview. A better approach is to use JavaServer Faces (JSF), which is part of the Java Enterprise Edition (Java-EE) [18, p. 9]. It uses Servlets and JavaServer Pages (JSP) to generate HTML pages [25, p. 121 ff]. Figure 4 illustrates how the page generation looks like on a Server. For the Peer-to-Peer framework it is important to understand, that the client and server process are running on the same machine. In step one, the client requests a *.jsp page. This page contains HTML code and JSF tags[6]. The server receives the request and loads the corresponding *.jsp file. After that the server generates a Java file out of this page. This class inherits from *javax.servlet.http.HttpServlet*, when the HTTP protocol is used. Simply spoken, the java class contains a method which returns the HTML page. The *.jsp content is copied into this method and all non HTML parts are replaced. In step four and five the generated Java class is compiled and executed. Finally, the server returns an HTML document to the client in step six. JSF itself is a

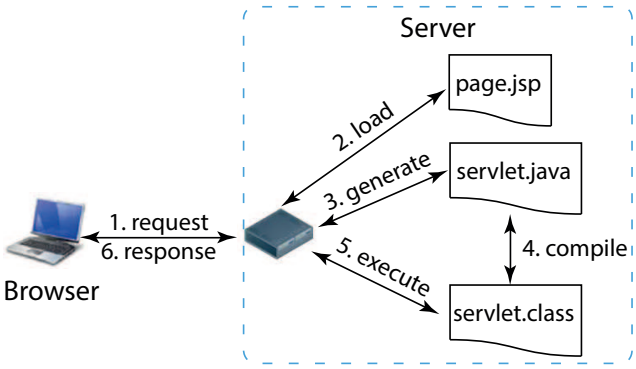


Figure 4: HTML generation
Source: Own figure based on [25, p. 71]

framework which contains components for the Graphical User Interface (GUI) of a web application. So it is comparable to Swing or JavaFX. It also allows to create new components [18, p. 9]. Moreover, it also manages the communication between client and server. The application layer is represented by so called *Beans*. *Beans* are simple Plain Old Java Objects (POJOs), which can be accessed from a *.jsp page. More concrete, when a *.jsp file is used to create a servlet, all necessary *Beans* are created as well. There are different kinds of *Beans*, which for example have different life cycles. This supports the storage of information as long as a session takes [25, p. 61 ff.]. An advantage of JSF is its separation of the application and presentation layer. Therefore, it is easier to work with multiple people on the same project. With JSF 2.0 a new standard was introduced. It is called Facelets and provides a page description language.

[6] JSF tags are XML based

Facelets defines server side objects, which are represented as tags [18, p. 11]. These tags will be translated into HTML code during a request. An example for this is <h:table>and <h:column>. Both will be translated into the HTML tags for a table. Newer IDEs uses a graphical editor, so it is not necessary to write the tags by hand. Insteads it is sufficient to add components on a drawing area and the JSF tags are automatically inserted into the jsp page. [18, p. 235]. This reduces the time needed for the designing precedure.

When the client receives the HTML page he needs a possibility to render it. This is done by using an appropriate kind of browser. The most obvious solution is to use a native web browser, which usually already come along with the operating system used. Today nearly every computer has such a program installed. Java allows to start a native browser with an opened web page. So the users do not need to do this manually, which makes this a feasible solution. One disadvantage is, that the users now need to switch between the browser and the P2P Framework application. Another problem is, that not all native browser support WebRTC. Not supported browser are for example the *Internet Explorer* and *Safari* [5]. At least for Windows there is a small workaround. The Project *webrtc4all* allows to use WebRTC with the most common browsers [22].

3.3 Java Browsers

Another idea to render HTML documents is to use a browser, which is directly integrated in Java. There exist some different libraries to achieve that. The most promising ones are the Eclipse Standard Widget Toolkit (SWT) browser and JavaFX. Eclipse SWT is an open source library for Java. It was introduced by IBM in 2001 and provides access to a set of native GUI components of the operating system. Eclipse SWT is available for the most common platforms (Windows, Linux and OSX) [30]. Besides components, like Buttons, Lists and more, there also is a browser component. This component is able to display HTML documents directly in Java. During the test the browser component was able to load a simple HTML page. The loading time of the page was quite fast and the structure of the page seemed to be correct, as well. The next part of the test was to load an HTML5 page. This page contains a <*video*> tag with a video file as a source. The Eclipse SWT browser played this video file smoothly, so it passed this part of the test. Another pleasing ascertainment was, that the needed CPU power was very low during the video playback. A small deficit was, that fullscreen mode is not supported. The last part of the test was an HTML5 page, which uses WebRTC to connect to the video camera and microphone. Unfortunately, it was not possible to start the video capturing. But Eclipse SWT provides the possibility to change the rendering engine. For example, it is possible to switch to WebKit. But this requires some adjustment. On a Windows system, for example, it is required

to use only the 32-bit SWT version, which also requires to use a 32-bit JVM. Additionally, Safari[7] has to be installed as a precondition [31]. However, that does not bring us towards a solution, since Safari does not support WebRTC. Another selectable rendering engine is a Mozilla based engine. But this requires the installation of XulRunner. XulRunner is a runtime package, which accepts the use of a Mozilla based engine in other programs. Therefore, it is usable for the Eclipse SWT. Nevertheless, it is difficult to get it running. Different versions of the operating system require a different version of XulRunner. Furthermore, the version of SWT and XulRunner need to fit too [31]. But even when all required settings are done, it does not mean that it works properly. The test on a Windows operating system ends with an error. SWT is updated continuously, so it is possible that this error will be fixed. The other promising technology is JavaFX, which was introduced in chapter 2.4. JavaFX contains a browser component, which was able to load a simple HTML page during the test. The loading time varied a lot depending on the operating system used. In sum, the browser was a little bit slower than the SWT competitor. However, it reaches an acceptable value. Due to the fact, that JavaFX supports HTML5, it also passes the test, where the web page contains HTML5 content. The video needed more time to load initially, but after that the browser plays the video smoothly. The loading time for this test also varies depending on the operating system.

The support for the fullscreen mode is missing in JavaFX as well as in SWT. JavaFX failed the vital test, where the connection to the video camera and the microphone was tested. The reason for that is, that JavaFX does not support WebRTC in its current version. But it could be possible, that this might change in the future. According to the roadmap of JavaFX, it will provide enhanced HTML5 support in the next version, which is announced for late 2013 [24]. But there is no clear announcement for WebRTC support. The big benefit of Eclipse SWT and JavaFX is that they allow to integrate a browser in Java. Therefore, the user does not need to switch to another application for the video communication. Another advantage is that the needed CPU power is much lower compared to the OpenIMAJ and OpenCV technology.

4 Collaborative Work

Another topic of this paper is the collaborative work on different media. But what does collaborative work mean? ”Collaboration is a working practice whereby individuals work together to a common purpose” [1]. Collaboration can be distinguished in two parts. *Synchronous* and *asynchronous* collaboration. *Synchronous* collaboration comprises features like online meetings and instant messaging. The former can be video chats, which was considered by the previous chapters. *Asynchronous* collaboration on the other hand considers shared workspaces and annotations [1]. The following part of this paper will consider both aspects.

4.1 Text documents

This chapter will introduce two possible technologies to work together on the same text document. This paper defines synchronous collaboration on a text document as follows: Every participant is aware of changes on a text document during a session. Therefore, it is important to transmit the changes at small delay. Asynchronous collaboration on the other hand means, that a user gets the newest version of a document when it is modified by another participant, during his absence. The first candidate is Brainlink. Brainlink is an open source software, which is developed at the University of Glasgow. It is based on the Network Text Editor, which is created at the University College London. Brainlink is written in pure Java, so it is available for all Java supported platforms. Brainlink describes itself as a collaborative text editor [34]. However, the description is kind of irritating. This is because Brainlink uses a drawing area. By clicking on this area the software creates a small text field. Only in this text fields it is possible to write text. Due to using a set of text fields, it is possible to change the order of them, by bringing them to the front or back. It is also possible to delete the text fields. Unfortunately, the movement of the text blocks, on the drawing area, is not supported. Another disadvantage is, that it is only allowed to write in a single line. Pressing the Enter button seems to add a new row in the text field, but it is not possible to write in this row. The interesting part is the connection of multiple Brainlink editors. One editor hosts a session and other editors can connect to this session. The host has an overview of all connected clients. He is also able to see the text fields and its content, which the connected clients create. Another pleasing feature is, that the host is allowed to manipulate the text inside this fields. Unfortunately, it is not possible to distinguish the text fields, so the host does not know which field is created by whom. Another problem is, that the connection only works in one direction. The clients can only see their own fields and they also do not get an update, when the host changes the content of one of those fields. It seems that this behaviour is intended. Therefore, Brainlink provides synchronous collaboration only in one direction, while asynchrounous collaboration is not supported. The last update of Brainlink was in 2004 [34], so it is improbable that the named disadvantages and problems will be fixed. Although it is an interesting approach, to encapsulate text into fields, the project is not suitable for the P2P Framework. A conceivable solution would be to extend Brainlink by removing the problems and provide bidirectional collaboration.

The other technology this paper considered is called DocShare. DocShare is a Plugin of the Eclipse Communication Framework (ECF), which ”provides a modular im-

[7] Safari is a webbrowser developed by Apple

plementation of the OSGi [...] Remote Serivice standard" [29]. In DocShare the participants can take on two different roles. The initiator or the receiver. A participant is an initiator when he shares a text file with another user, which is called the receiver. To do that both participants have to be connected to a provider, which is supported by the ECF Datashare API (XMPP, XMPPS, etc.) [28]. The initiator can now invite a user through this provider. The receiver gets an invitation and has the possibility to reject or accept it. After that the receiver has a copy of the initiator's editor and content. From now on both participants can work simultaneously on the same document [28]. DocShare uses a special algorithm, called *Cola* (abbr. for collaborate) [11, 00:17], to achieve an appropriate behaviour during the editing process. From this it follows that DocShare supports synchronous collaboration. Asynchronous collaboration is not fully supported. Both participants will have the same document when their mutual session is closed. But then these documents are not linked together anymore. Therefore, when a new session is started it is handled like it is a new document. Another disadvantage of this approach is, that it only supports two participants, which can work simultaneously on only one document. Otherwise it provides a good behaviour while editing. For example, DocShare can handle a situation where one participant deletes a whole section while the other user is inserting text into that section. This situation is solved by splitting the deletion [9, 46:47]. Given the fact that ECF can be used in a Eclipse RCP application, it is a suitable solution for the P2P Framework. However it is necessary to implement a provider, which can be supported by ECF Datashare API. It is also thinkable to modify the framework by adding support for more than two participants and more documents. This is possible, because according to the author of Cola, the algorithm does not need to be changed to achieve that [9, 32:55].

4.2 Calendar

Another nice to have feature for the P2P Framework would be a calendar. A good solution is to build an own version of a calender application for the P2P Framework. That does not mean that it is necessary to write it from scratch. There are several technologies which can help. One piece of the calender application could be the iCalendar standard. It is a file format to exchange calendar information. It contains information like the author, description, time, etc. [4]. There exist several libraries, which make it possible to use iCalendar in Java. One of them is iCal4j [7]. The second piece is a visual representation. Usable ones are NachoCalendar [16], JavaFX Date Picker [26] or migcalendar [17]. The last missing pieces are a database where the iCalendar information can be stored and an interface between these database and the representation of the calendar. Additionally to consider is, how the information can be shared. One possible scenario is, that the owner can only modify his own calendars and is able to share them with other users. These users can only read the information on these calendars. Another approach would be that multiple users can modify the same calendar, but this demands much more effort for the implementation. An algorithm similar to Cola is needed to achieve such a behaviour. For the first approach, synchronous and asynchronous collaboration can be achieved, by updating the last changes of a calendar as soon as possible. For the second approach a similar definition like for text documents applies.

4.3 Draw Boards

The last media type for collaborative work in this paper is a draw board. This section will give a small overview over some technologies which allows to draw together on a drawing area. For collaborative drawing the same definition of synchronous and asynchronous collaboration as for text documents applies. One candidate is Collaborative Drawing. It is developed by Umang Mathur at the Indian Institute of Technology Bombay. Collaborative Drawing is written in pure Java and uses a client-server architecture [15]. Both participants, client and server, have a canvas, on which they can draw free handed. The drawn content is transmitted directly to the other participant. Therefore, Collaborative Drawing supports a synchronous collaboration. Asynchronous collaboration is not supported, because the content only exists as long as the session is open. The advantage of Collaborative Drawing is that the drawn content is transmitted very fast. But there are also some disadvantages. One of them is that the software only allows one kind of drawing. This means, that it is only possible to draw free handed and it is not possible to change the colour, thickness or other features. A much bigger disadvantage is the fact that Collaborative Drawing needs a lot of CPU power. Even if no participant is drawing. This makes it only partially usable for the P2P Framework. Another candidate is Amity-Drawing, which is an open source application. It works similar to Collaborative Drawing, but due to the fact, that it is incomplete and the last changes are from 2012, makes it not suitable for the P2P Framework [19].

A better solution would be to use a drawing application for Java and implement the collaborative behaviour. A possible candidate would be Steve's Mediocre Paint, which is an application similar to Microsoft Paint written in Java [35]. The implementation of a collaborative behaviour is a complex task. It requires techniques similar to those in Cola. Especially the erasing of content, while someone other is adding something in the same area efforts a special procedure. Another approach would be to use layers. Similar to the technique Brainlink is using in chapter 4.1. This means, that every participant is drawing on his own layer or layers and the other users are only able to see this changes but are not allowed to modify it.

5 Summary and conclusion

The aim of this paper was to provide an overview of available technologies for audio, video capturing and for collaborative work. Therefore, different approaches were presented. One of them was the integration of multimedia content directly in Java. It quickly became clear that many technoloiges, like JMF, FMJ and LTI-CIVIL, are too old and are not improved further. Besides these solutions, some technologies focus on more complex problems such as face recognition or image manipulation. To achieve that, they provide access to video cameras. OpenCV and OpenIMAJ are the most promising solutions in that area. In my opinion, OpenIMAJ is more attractive, because first it needs less disk capacity than its competitor and second it does not need a separate installation to provide its features. A small disadvantage of both technologies is, that they produce more CPU load compared to the other aproaches. But the calculated values are acceptable. One problem that needs to be solved is the amount of data generated during the streaming. Transmitting pictures uncompressed is not feasible. So the project group has to implement one of the solutions I provided in chapter 2.2. Another disadvantage is, that it is necessary that the libraries are held up to date. Otherwise, hardware changes and new operating systems can

Table 1: Technologies - advantages and disadvantages

Java integrated technologies			
Technology	**Feasible**	**Advantages**	**Disadvantages**
JMF	×	-	• Obsolete • No webcam support
FMJ + LTI-CIVIL	×	-	• Obsolete • Webcam frame rate very slow
OpenIMAJ	√	• Integrated in Java • Features like face recognition • Embedment of jars sufficient • Mobile devices: Android	• Data needs to be compressed for streaming • Higher CPU load
OpenCV	√	• Integrated in Java • Features like face recognition • Mobile devices: Android & iOS	• Separate installation needed • Many disk space required • Data needs to be compressed for streaming • Higher CPU load
JavaFX	×	• Future-oriented replaces Swing, AWT and is part of Java 8	• No webcam support yet
HTML5 - WebRTC			
Technology	**Feasible**	**Advantages**	**Disadvantages**
Native Browser	√	• Installed on nearly every computer • Often updated • Future-oriented mobile browser support	• Not all browsers support WebRTC yet • User needs to switch to browser
Eclipse SWT	×	• HTML5 already supported • Integrated in Java • Native rendering engines available	• No WebRTC support yet • very choosey (OS and JVM - 32 or 64 bit) • Rendering engines complex to install
JavaFX	×	• HTML5 already supported • Integrated in Java • Future-oriented replaces Swing, AWT and is part of Java 8	• No WebRTC support yet

make these approaches unusable. A promising technology is JavaFX. JavaFX seems to play a more central role in the future. Not only for the fact that it will be fully integrated in the next Java version, but also that it seems to replace the much older GUI related technologies like Swing and AWT. Therefore, it is possible that a support for video capturing is coming. Unfortunately it is a little bit risky to wait until late 2013, because there is no statement that this support will be provided in the next Java version. Another potential solution is to use HTML5 and therefore the WebRTC standard, which was presented in chapter 3. Particularly pleasing is, that this solution establishes its own peer-to-peer connection, so it is not necessary to implement the compression or the streaming. To present the multimedia content this paper introduced two approaches. One solution is to use a native browser. On the one hand it is great that these browsers are mostly up to date, so the access to the hardware is more assured. On the other hand WebRTC is quite new and not even finished. Therefore, there are only a few browsers which support this standard. The same applies for the browsers, introduced in chapter 3.3, which can be integrated into Java. Eclipse SWT looks promising but it is very choosey, referred to the used operating system and JVM version (32bit or 64bit). The other problem is that it does not support WebRTC yet. JavaFX seemed to be the better solution, but liked mentioned in chapter 3.3, there is no support for WebRTC as well. Furthermore, the use of WebRTC also needs some implementation work. On the one hand it is necessary to generate the HTML page which can be represented in the browser. JSF, which was introduced in chapter 3.2, seems to be a good idea for that. On the other hand WebRTC needs a STUN server. I am not sure if WebRTC accepts a self made server on the localhost. The most important advantages and disadvantages, of the introduced technologies, are summarized in table 1.

In the last chapter this paper introduced possible solutions to work collaboratively on different kinds of media. The most important question is, how these media should be handled. Particularly referring to the synchronous and asynchronous behaviour, the Cola algorithm seems to be a good way to handle it in text documents. How this can be translated for the other kind of documents, need to be discussed in the project group. Maybe it is not necessary. At least for the calendar, I prefer the solution where every participant have only the rights to modify his own calendars. In conclusion it can be stated that each technology has its advantages and disadvantages. In any case it will be a challenging task for the project group.

References

[1] AIIM Global Community of Information Professionals. What is collaboration? http://www.aiim.org/What-is-Collaboration. (last checked 2013.04.29).

[2] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

[3] ConnectSolutions. Xuggler. http://www.xuggle.com/xuggler. (last checked 2013.04.29).

[4] B. Desruisseaux. Internet Calendaring and Scheduling Core Object Specification (iCalendar). http://tools.ietf.org/html/rfc5545, 2009. (last checked 2013.04.29).

[5] S. Dutton. Getting started with WebRTC. http://www.html5rocks.com/en/tutorials/webrtc/basics/, 2012. (last checked 2013.04.29).

[6] H. M. Eidenberger and R. Divotkey. *Medienverarbeitung in Java - Audio und Video mit Java Media Framework und Mobile Media API*. dpunkt.verlag, 2004.

[7] B. Fortuna. iCal4j. http://ical4j.sourceforge.net/. (last checked 2013.04.29).

[8] Google. WebRTC. http://www.webrtc.org/. (last checked 2013.04.29).

[9] Google Tech Talks. Eclipse day at the googleplex: Wiring hacker synapses. http://www.youtube.com/watch?v=GfeUCT-tRJQ, 2008. (last checked 2013.04.29).

[10] J. S. Hare, S. Samangooei, and D. P. Dupplaw. Openimaj and imageterrier: Java libraries and tools for scalable multimedia analysis and indexing of images. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 691–694, New York, NY, USA, 2011. ACM.

[11] M. K. Isik. Cola: Real-time shared editing. http://vimeo.com/1195398, 2008. (last checked 2013.04.29).

[12] itseez. About opencv. http://opencv.org/about.html. (last checked 2013.04.29).

[13] itseez. Opencv. http://opencv.org/. (last checked 2013.04.29).

[14] Larson Technologies Inc. lti-civil. http://lti-civil.org/. (last checked 2013.04.29).

[15] U. Mathur. Collaborative drawing. https://github.com/umangm/Collaborative-Drawing. (last checked 2013.04.29).

[16] I. Merani. NachoCalendar. http://nachocalendar.sourceforge.net/. (last checked 2013.04.29).

[17] MiG InfoCom AB. Java calendar, the mig java calendar component. http://migcalendar.com/. (last checked 2013.04.29).

[18] B. Mller. *JavaServer Faces 2.0: Ein Arbeitsbuch für die Praxis*. Carl Hanser Verlag GmbH and Co. KG, 2010.

[19] Open Source Community. Amity-drawing. https://code.google.com/p/amity-drawing/. (last checked 2013.04.29).

[20] Open Source Community. Fmj project. http://fmj-sf.net/. (last checked 2013.04.29).

[21] Open Source Community. Javacv. https://code.google.com/p/javacv/. (last checked 2013.04.29).

[22] Open Source Community. webrtc4all. https://code.google.com/p/webrtc4all/. (last checked 2013.04.29).

[23] Oracle. Java archieve downloads. http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-java-client-419417.html#7372-jmf-2.1.1e-oth-JPR. (last checked 2013.04.29).

[24] Oracle. Javafx roadmap. http://www.oracle.com/technetwork/java/javafx/overview/roadmap-1446331.html. (last checked 2013.04.29).
[25] Prof. Dr. G. Engels. Technologies for web-based systems. http://is.uni-paderborn.de/fileadmin/Informatik/AG-Engels/Lehre/WS1213/WE/slides/WE-2012-04.pdf, WS 2012/13. (last checked 2013.04.29).
[26] C. Schudt. My javafx calendar control. http://myjavafx.blogspot.ch/2012/01/javafx-calendar-control.html. (last checked 2013.04.29).
[27] The Arcomem Consortium. Openimaj. http://www.openimaj.org/. (last checked 2013.04.29).
[28] The Eclipse Foundation. ECF/DocShare Plugin. http://www.eclipse.org/ecf/. (last checked 2013.04.29).
[29] The Eclipse Foundation. Eclipse communication framework. http://www.eclipse.org/ecf/. (last checked 2013.04.29).
[30] The Eclipse Foundation. Eclipse swt. http://www.eclipse.org/swt/. (last checked 2013.04.29).
[31] The Eclipse Foundation. The swt faq. http://www.eclipse.org/swt/faq.php#howusewebkit. (last checked 2013.04.29).
[32] C. Ullenboom. *Java ist auch eine Insel*. Galileo Computing, eight edition, 2009.
[33] C. Ullenboom. *Java ist auch eine Insel*. Galileo Computing, tenth edition, 2012.
[34] University of Glasgow. Brainlink. http://brainlink.sourceforge.net/. (last checked 2013.04.29).
[35] S. Vitali. Steve's mediocre paint. http://sourceforge.net/projects/mediocrepaint/. (last checked 2013.04.29).

Peer to Peer Framework : Secure replication

Rohan Sukumaran
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email: srohan@mail.uni-paderborn.de

Abstract

The security of data in any available network is always important as any other factor in the network. For peer-to-peer networks the data security becomes even more important as the network itself is decentralized. Unlike the centralized networks where high-availability, genuineness and correctness of data and the network can be guaranteed, the peer-to-peer networks are formed by nodes or peers that can leave the network at any point of time without any notifications or warnings. To overcome these issues, data is replicated among the peers in the network so that the high-availability of data can be achieved even though the peer responsible for the data may leave the network. These data replications should be done carefully and secure mechanisms has to be incorporated so that the genuineness and correctness of data can be preserved.

This paper investigates the effect of malicious peers in replication of data in the structured peer-to-peer network and also gives an overview of the security issues in data storage, the possible methods to overcome these threats on secure data storage in the network and other available secure storage mechanisms for peer-to-peer systems.

1 Introduction

De-centralized and distributed networks are becoming more popular day by day. Peer-to-peer networks are a type of distributed networks where there is no central point of authority. Having such networks has its own advantages and disadvantages. Since there is no centralized system, failure of a node does not bring down the network as is the case of networks which have a central point of authority.

The challenges in the distributed and decentralized networks like the peer-to-peer networks, would be the genuineness and correctness of the data. Since the nodes in a peer-to-peer network, known as peers, are distributed over the network and the non-existence of centralized system to verify the correctness and genuineness of the data, its always difficult to know whether the data presented by a peer is correct and genuine. Security of the data in the network has to be taken care, which includes integrity, availability of data even though there could be possible existence of malicious peers. Malicious peers in network can cause the network itself to come down, also cause a lot of security issues in the network as well as in the data storage and routing information in the peer-to-peer networks.

To overcome these security issues, data replication, secure storage mechanisms, access controls and encryption of the data in the network are required. The security mechanisms employed should provide methods to protect the data from attacks and provide identity protection mechanisms or anonymity [1] and also the effect of malicious peers on data and routing in any peer-to-peer networks must be minimized.

This paper first looks at the attacks that can occur on the data in section 2. It's followed by replication strategies that are implemented in peer-to-peer networks in section 3 to ensure high availability of data in the network, avoid single point of responsibility which helps to counter storage and retrieval attacks [2], minimize latency in data access and distribute the load in the peer-to-peer networks and finally the security mechanisms that are available to prevent the attacks on the storage and replication of data in section 4 which can be implemented to form sophisticated security mechanisms that can protect the data and guarantee the genuineness and integrity of data.

2 Storage attacks

Replication of data in the peer-to-peer network is done so that the data becomes available in the network even when the responsible peer for the data becomes unavailable in the network. In the replication of data in network malicious peers can also participate. This participation can cause a lot of security issues on the data replicated in the network. We take a look at some of the attacks on storage available in peer-to-peer network and the effects of malicious peers in replication of data through the network.

2.1 Storage and retrieval attacks

Storage and retrieval attacks [2] is among the common attacks that can occur on the data. In peer-to-peer networks data is replicated among the peers to ensure data availability and performance in the network. Malicious peers can also participate in this replication of data, unless its detected by the security mechanisms present to detect whether that the peer is malicious or not. The malicious peers can correctly participate in the replication and lookup of data in the network [2]. But the malicious peer can provide false information to other peers on the replication and the data present in the network, deny request to the data.

The malicious peer upon receiving the request can deny the data to the peer even though it was the peer that was responsible for the data [2]. This can lead to loss of data present in the network. Another possibility would be that the malicious peer can participate in replication of data in the network and deny the access to replicated data to any other requesting peer in the network [2]. This can cause overload on any other peer in the network as the good peers in the network may be overloaded with the request to the particular data. This overload on the peer can lead to denial of service attacks in the network. To overcome these issues the replication strategy in network should be implemented so that the peer responsible for data should not be in charge of data replication in the network. Else the malicious peers can choose not to replicate the data and deny the existence of data. This leads to the loss of data in the network. The responsibility of replication of data in the network should be shared among peers, i.e. the replication strategy implemented should not allow single point of responsibility [2] in the network.

The malicious peers can insert the junk data into network [1]. This can cause problems with the integrity of data available in the network . During replication of data, the inserted junk data is replicated among another peers causing a lot of wastage in the storage available. These attacks are termed as poisoning attacks. To overcome this, quota limits can be employed [1]. For each peer in the network, quota limits can be imposed so that malicious peers cannot insert or replicate large amounts of junk or unwanted data in the network.

To minimize this attack on the network identification of peers in the network is always important. This can help in verifying the authenticity of the peer in the network. This in-turn helps in identifying and isolating the malicious peers in the network and minimize the effect of storage and retrieval attacks. The following subsection gives a brief explanation on the specific type of storage and retrieval attack called poisoning attacks.

2.2 Poisoning attacks

Poisoning attacks cause issues with the integrity of data in the network. In poisoning attacks, the attackers or malicious peers presents junk or unwanted data to any other peer in the network. This causes issues with the data stored on the network and wastes a lot of storage space in the network. In this section we take a look at the common poisoning attacks in peer-to-peer networks.

2.2.1 File poisoning

File poisoning attacks are done by copyright holders to prevent the distribution of copyrighted files in the network [3]. Copyright holders are corporations that own exclusive rights on the distribution of a file. File poisoning attack will cause problems with the censorship free data sharing in the network. One way is insert large amount of junk data into the network called decoys [3]. The decoys match the original meta-data information about the data, but the original data is complete garbage or useless [3]. This attack is called as item poisoning [3]. The peers in the network can accidentally add certain information or can encode the file in such a manner that certain copies of the file becomes unusable. This is known as network pollution [3]. In this attack the DHT (Distributed Hash Tables) indexes are not poisoned, only the file with any peer in the network is poisoned i.e. the file contains valid meta-data information but the file with the peer is useless.

This particular attacks can also be used by any malicious peers in the network or attackers that want to make the data in the network as unusable. The malicious peers can insert the junk data into network with the same meta-data information, the item poisoning attack [3]. Any lookup for the data in the network will present this garbage data, even though all the meta-data information of the data was correct and genuine. One way to stop this from happening is to use quota in storage so that only a certain portion of the network storage contains junk data. This factor highly depends on the amount of the malicious peers in the network.

2.2.2 Index poisoning

Index poisoning attacks are also done by the copyright holders to prevent the distribution of copyrighted files in the network [4]. In this attack the malicious peers or attackers poison the index in the DHT (Distributed Hash Tables) for a set of particular data D [4]. The index is made to point to the decoys [3] or the bogus data in the network [4], which includes the bogus data, peer id's, file identifiers and IP address [4]. In file poisoning attack described in section 2.2.1, the DHT indexes are not poisoned, the data D with any peer is poisoned, unlike the index poisoning attack.

The lookup for the particular data D by any peer in the network, the index in the DHT return these decoys or the bogus records that were inserted by the malicious peers in the network. The requesting peer will download this data, which is junk and useless. Since the index's are poisoned,

the peers cannot distinguish easily between the correct, genuine data and the decoy or polluted data in the network [4].

2.3 Routing attacks

Routing attacks cause major problems in any network. The malicious peers in the network can redirect the requests for data so that the requesting peer never gets the data it was looking for. It can also cause problems in replication of data in the network. The effects of this attack on routing and network overlays are not considered in this paper. In this section we take a look at the two of the common routing attacks and how these attacks cause problems for the good peers in accessing data and replicating the data in the network.

2.3.1 The Sybil attack

Sybil attack [5] is one of the common routing attacks that can occur in peer-to-peer networks. In Sybil attack a malicious peer can masquerade under different identities in the network. This causes problems with routing and in data access and storage in the network. The masquerading peer can participate in the replication of data in network, it can provide false information on the data replication because the malicious peer takes different identities, the replication algorithms or strategies, discussed in section 3 will assume that enough replicas has been created for the data. When the request to the data arrives, the malicious peer can deny the existence of the data. Subsequently the malicious peers can also insert large amount of junk data, place decoys in the network causing pollution and poisoning [3] [4] in the network, described in section 2.2.

2.3.2 Eclipse attack

Eclipse attack [6] is another type of routing attack that occurs in the peer-to-peer network. Sybil attack [5] can be used to perform eclipse attacks [6] on the network. In eclipse attack, a group of malicious peers in the network can create problems for the good peers in the network by creating a sub-network in the peer-to-peer network. All good peers in the network having neighboring sets as this group of malicious peers will not be able to get any resources from the network. Since the requests has to pass through any one of the malicious peers, the malicious peer will either drop the request from the good peers, or it can send a message to the requesting peer indicating the non-existence of the requested resource. Subsequently the malicious peers can cause network pollution and provide junk data on the request from other peers causing poisoning attacks, described in section 2.2. These group of peers causes the eclipse attack [6]. In this method some of the peers in the network is "eclipsed" or hidden from the rest of the networks by the group of malicious peers. This way the group of malicious peers partitions the network into multiple sub-networks. Figure 1 shows how a group of malicious peers (peers with the warning icon) performs a coordinated attack to divide the network into three sub-networks.

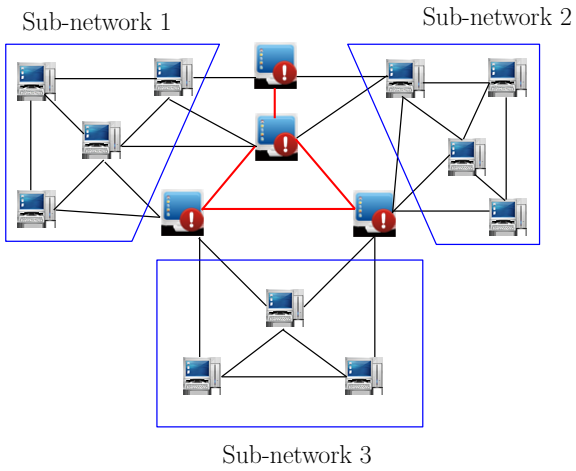


Figure 1. Eclipse attack in which the malicious peers dividing the network into three sub-networks.

Source : Own figure based on eclipse attack [6].

3 Replication in storage

Any network that is available in the present world have an underlying storage mechanism to store the data, so that it can retrieve the data and present the data to any request it encounters for the data. The storage also guarantees that the data is available in case of power loss, i.e. non-volatile storage mechanism. In centralized networks, the storage in the server can be easily be managed by the operating system running on the server. In case of peer-to-peer networks the storage spaces are contributed by the peers in the network and it employs a distributed approach to store the files. The storage mechanism in peer-to-peer network relies on DHT (Distributed Hash Tables) [7] for managing the data in the network. DHT can be considered to be similar to Hash Tables [7]. It stores data in $\{key, value\}$ pairs similar to Hash Tables. Any peer in the network can retrieve the value associated with a given key from the DHT. The hash function maps this key to the index in the DHT and provides information on the location of data in the peer-to-peer network.

In peer-to-peer networks files are replicated in the network so that the file or data becomes available to any other requesting peer when the peer responsible for the data becomes unavailable or leaves the network. Replication enables the high-availability of data through the network and avoids the single point of responsibility or failure in the network [2]. In this section we take a look at the different avail-

able replication strategies that are used in the peer-to-peer storage mechanisms.

3.1 Passive replication

Passive replication [8] is one of the simplest form of replication strategies which can be employed in the peer-to-peer network. In this strategy the file is replicated upon request by any other peer to the file or data.

Consider a peer-to-peer network where there are n number of peers. Peer i holds the file and if another peer j requests the file, the file is copied to the peer j from peer i. Now in the network we have 2 peers having the files; peer i and peer j. Now if any other peer k request for the file, it would have 2 peers i and peer j from which it can retrieve the file. The replication of data in network is carried in such a manner, i.e. the files are replicated only upon request from the other peers in the network. When the data in the network gets updated, the other peers in the network has to check with the peer responsible for the data. Since this method does not maintain any associated list, the algorithm should be modified to accommodate the update of data.

In such replication strategy the replication of the file depends on its popularity, as the most popular file would be found in almost all the peers in the network. Another problem that could occur is that the peer responsible for a particular file goes off-line or becomes unavailable in the network, and the file was not requested by any other peers in the network, then the data becomes unavailable.

3.2 Active replication

Active replication [7] is also known as proactive replication [9]. This replication strategy is employed to guarantee the high availability of data in peer-to-peer networks. Unlike passive replication [8], this method aims to replicate data to all other peers in the network so that if any request for the data is encountered in peer-to-peer network with n peers, there are large number of peers in the network with data that can serve the request.

An example of such implementation is Tempo [9]. Tempo is a DHT (Distributed Hash Tables) [9] in which the replication is carried out at regular time intervals in the network unlike any other replication strategies that relies on data request or node failure for replication of data in the network [9]. Tempo does not take care of the data update operation. The implementation should be modified to incorporate the data update process.

In this replication strategy the popularity of file is not affected in replication of the file through the network. This method helps to guarantee the high availability of data in the network. This can also help in reducing the latency to access data and balance the load in the network with large amount of peers holding the data.

3.3 Cache-based replication

Cache-based replication[8] strategy employs the method of storing the data in the peer as it passes through the peer in the network. This replication strategy are normally employed by many peer-to-peer storage mechanism alongside with other replication strategies. Cache-based replication [8] is employed to increase the performance of the network by decreasing the latency in getting the file.

An example would be the caching of data in PAST [10]. In PAST [10] cache management is done to achieve minimum latency in file access [11] and to achieve other certain goals — to maximize query throughput [11]; to balance query load [11]. The replication algorithm in PAST replicates any data among k neighboring peers for a given peer in the network. When a request for data in the network occurs, the data can be served from one of the replicas or from the peer responsible for the data. When the data is routed to the requesting peer, PAST [10] inserts the routed data into a peer *p1's* cache if the size of the data is some fraction less than the peer *p1's* cache size [11]. For any other request to the data item that is routed through peer *p1*, the data is send to the requested peer from *p1's* cache. This reduces the latency in accessing the requested data. When the data in the network gets updated, the data in the peers cache gets invalidated and the whole process repeats.

This strategy can be useful also for balancing the load in the network because any set of peers wouldn't be overloaded when the requested file is very popular. In such cases the caching of file among the peers would considerably reduce the load on a cluster of peers in the network. As a result of caching, it helps in reducing the latency in data access in the network.

3.4 Dynamic replica management

Dynamic replica management strategy [8] goal is to achieve minimum replication in the network and provide better service. In this strategy the replicas are minimized, latency to access data is minimized, load is balanced in the network and ensure high data availability.

Replica Enumeration (RE) [12] is a technique that implements dynamic replica management. In this technique when data is requested by a peer, a decision is taken from the list of available replicas by determining the closest available replica to the requesting peer. The closest replica is chosen to serve the request to reduce the latency in data access. In case of an overloaded peer in the network, the overload peer may return a list where the requested data is cached or a list other known replicas is returned [12]. From this list the closest replica or peer is chosen to get the data. When the data in the network gets updated, the replicas are invalidated. New replicas are placed with the updated data.

The dynamic replica management algorithms [8] employs this technique of searching replicas with minimum

latency. If the replica or peer is overloaded, then it searches for another replicas with next minimum latency and so on. If the replicas found have latency more than specified by the requesting peer, then these algorithms try to create new replicas closer to the requesting peer thereby distributing the load on the network and try to minimize the latency in access of data in the network.

3.5 Introspective replica management

Introspective replica management strategy [8] relies on the network traffic to determine whether the replication of data is to be carried or not. The introspection model is implemented in OceanStore [13] for replica management in the network. Here the system observes the activities on the network and adapts to the situation to provide better service.

Introspection model implemented in OceanStore [13]. It consists of two main modules :

- The observer module [14]
- The optimization module [14]

The observer module [14] records the activities in the network and records many parameters, these parameters are analyzed to extract patterns that will be used by the optimization module [14] which performs optimization in the network by adaptation [14]. In case of replication in the network, the replicas or peers adapt to these patterns by replicating the data and placing additional replicas thereby optimizing the latency in access of data and distributing the load in the network. The additional replicas created can also be terminated [13]. If the replica is found to be not useful, the optimization module can terminate the replica freeing up space for other data that can be useful to optimize the network. When the data in the network gets updated, the replicas are invalidated. New replicas are placed with the updated data based on the introspective technique.

The introspective replica management technique tries to maintain the right amount of replicas in the network and tries to optimize the latency in data access and balance the load on the peers in the network. In OceanStore [13] it's also used for cluster recognition [13]. OceanStore [13] uses this technique alongside the Cache-based replication [8] strategy described in section 3.3 for better performance in the network.

3.6 Erasure coding

Erasure codes are used for communication of data [15]. These are forward error correction codes [15] that can be used to recover data without any re-transmission of the data. Hence they are implemented in fault-tolerant systems. In peer-to-peer networks erasure codes are used for replicating and to secure the data which is discussed in section 4.5.

4 Security in peer-to-peer storage

To overcome the effects of malicious peers on the network and the attacks that can occur on the data in the network, seen in section 2, the security mechanism has to implemented. These security mechanisms should ensure the integrity of data. The data availability can be managed through replication in the network as seen in section 3. In this section we take a look at the different security measures that can be useful to minimize the effects of malicious peers and attacks on data in the network.

4.1 Self-Certifying data

Self-certifying data [8] is one of the simplest form of data integrity checks that can be enforced in the network. With self-certifying data a peer that requests the data can verify the integrity of the data by itself. While inserting data in to the network, the responsible peer can use the known hash function to calculate the cryptographic hash of the file based on its contents [8] [16]. This technique is employed in CFS [17], while PAST [10] uses this approach alongside with signed files [11] [16].

When a peer request for particular data, the requesting peer can use the same hash function to calculate the cryptographic hash and match the hashes to check data integrity. However the technique is only useful if the requesting peer knows the hash for the data [16]. This method also requires a common hash function to be used in the entire network as the peers must use this hash function to calculate the cryptographic hash of the data in the network.

4.2 Information dispersal algorithm

Information dispersal algorithm [18] can be considered similar to erasure codes [15]. Information dispersal algorithm [18] is used in Mnemosyne [19].

In this technique the data of size S is transformed into n blocks. The data can be reconstructed from m blocks where m is less than n with a redundancy of m/n. The difference $k = n - m$ is the maximum failure that can occur in the network. In the network if more than k peers fail, then the data cannot be fetched. For efficient dispersal of the data, the k must be wisely chosen with the ratio $m/n \simeq 1$ [18].

4.3 Shamir's secret sharing scheme

Shamir's secret sharing scheme [20] is also known as *(k,n)* threshold scheme. This is implemented in many secure storage mechanisms, also can be found implemented in PAST [8] [10]. The *(k,n)* threshold scheme says that when the data D that is divided into $D_1, D_2, ..., D_n$ then :

- knowledge of any k or more D_i fragments makes the data D computable [20]
- knowledge of any *k-1* or fewer D_i fragments makes the data D uncomputable [20]

The *(k,n)* threshold scheme [20] is used for managing the cryptographic keys that encrypts the data. When k=n, then all the participants are required to construct the data [20]. Any peer in the network inserting the data into storage encrypts the data with a cryptographic key C_k. This key is then stored using the *(k,n)* threshold scheme [20]. The cryptographic key C_k is divided into $C_{k,1}$,$C_{k,2}$,...,$C_{k,n}$. A requesting peer now can get the key C_k, if it has the knowledge of at-least the k fragments of the n fragments of the cryptographic key C_k. Using this the requesting peer can decrypt the cipher and get the original data. This scheme is also used in secure decentralized erasure codes [21] which uses homomorphic public key encryption, described in section 4.5. For the data to be lost in the network, an attacker must gain control over $\{n\text{-}k\text{-}1\}$ peers that stores the n-divided cryptographic key C_k.

4.4 Smartcards

Smartcards are employed in PAST [10] for security of data and the peer in the network. These smartcards are held by each peer and user in the network [10]. These smartcards contains public/private key which are used to generate peer id's and certificates used for data operations. These smartcards also helps in managing the storage quota limits for each peer in the network [10] so that a malicious peer cannot overload the network with junk data. This storage quota enforced helps in achieving scalability and efficiency in the network.

In PAST [10], the peer id for a peer is generated using cryptographic hash on the smartcard public key of the peer [10]. This helps the peers in the network to verify the other peers peer id, thereby verifying the authenticity of the peer. Smartcards also generates file certificates when a peer wishes to insert data in to the network. The file certificate contain the cryptographic hash of the data, the file id, replication factor which are signed by the smartcard card [10]. This file certificate is used to ensure the integrity of the data. The data in the network can also be encrypted using any of the cryptographic encryption schemes to ensure the data privacy in the network [10]. This encrypted data can then be inserted into the network, using smartcards the integrity of the encrypted data can also be ensured.

Using smartcards requires that the peers need to obtain the smartcards, replace the smartcards periodically [10] as the smartcards key can be compromised, the keys used can get expired or the smartcard itself can be compromised by the attackers. To ensure the security in the network, smartcards has to be replaced regularly so that there is freshness in the keys [10] used for securing the data and the peer in the network.

4.5 Erasure coding

Erasure codes [15] are used for communication of data. These are forward error correction codes [15] that can be used to recover data even though the data may be corrupted. In erasure coding the data is divided into k different blocks [15]. These k blocks are transformed into n different data units. These transformed n data units are much larger than the k blocks. The original data can now be recovered from the subset of the n data units. The ratio *k/n* indicates that, of the n bits of data, k bits contain useful information which can be used to recover the data [15].

In replication on peer-to-peer networks, reperasure (replication with erasure code) protocol [22] can be implemented to ensure data replication in the network and to guarantee high data availability. During replication the n blocks of the file generated by erasure coding is replicated in the network and stored among the peers. Upon request to the data, only a subset of the n blocks are required by the peer and it can construct the original file using the erasure codes. The blocks that are created by the erasure codes are much lesser in size compared to the data which helps in saving the disk space in the storage during replication [22].

Decentralized erasure coding [23] enforce better security on the replication and storage of data. Unlike the normal erasure codes used in reperasure [22], the k different file blocks are transformed to n blocks independently using random linear coefficients [23]. The first block is formed by using k linear random coefficients on the k blocks of file. This is continued for the second block, third block and so on to get n transformed file blocks. These transformations are done by the peers in the network. This finally creates an $[nXk]$ matrix, which are also known as sparse generator matrices [23]. To decode the data a system of linear equations is solved for k different equations with k unknown variables using the $[kXk]$ matrix from the $[nXk]$ matrix that was generated during the transformation of the file to n blocks [23].

Secure decentralized erasure codes [21] which uses cryptographic encryption schemes [21] ensure better security than decentralized erasure codes. Homomorphic public key encryption is used for encryption [21]. The encryption scheme allows specific computations on the cipher. In this case it's done using a variant of the decentralized erasure codes. The modification on the cipher forms multiplicative homomorphic cipher [21]. The assumption made is that the key servers are much more secure than the peers in the network [21]. The key servers are distributed in the network. The key sharing can be implemented using secret sharing mechanisms [21], an example described in section 4.3. The key share holder's can then perform partial decryption of the data and the requesting peer can recover the original data from the partial decrypted data [21].

The erasure codes are forward error correction codes [15], modification of the data blocks or poisoning attacks, described in section 2.2 by malicious peers can also be detected by the requesting peer. Using the reperasure protocol [22] alongside with secure decentralized erasure codes [21]

and efficient cryptographic key management schemes, the availability, security and integrity of the data can be ensured in peer-to-peer networks.

4.6 Access control mechanism

Access control mechanisms can also be used for the security of the data. Access controls describe the rights of all the users in peer-to-peer network on the data. This technique is used to prevent the unauthorized access to data. Access control lists provides a mapping of the access rights for users to the data in the storage. In the access control technique discussed, the peers export their access control policies in XML format, called as eXtensible Access Control Markup Language (XACML) [24]. This XACML [24] is implemented using two approaches in peer-to-peer networks.

```
<Rule RuleId="export:p" Effect="Permit">
  <Subject>
    <AttributeValue DataType="&datatype;
        rfc822Name">user1</AttributeValue>
  </Subject>
  <Resource>
    <AttributeValue DataType="&xml;anyURI">d1</
        AttributeValue>
  </Resource>
  <Action>
    <AttributeValue DataType="&xml;string">read</
        AttributeValue>
  </Action>
</Rule>
```

Listing 1. XAML export policy granting user1 permission to read data id d1 on peer p [24]

The first approach is to provide a mapping of the XACML exported by the peers [24]. For this approach it's required that all the peers in the network generates the XACML file in a common format so that the mappings can be easily established. However this mapping is only done to the local data available with the peer *p*. So with this mapping, the other peers are allowed to access the data residing in peer *p*. This access control mapping has two levels of administration [24]. The first level, states which local users of the peer *p* are allowed to access the peer-to-peer network [24]. The second level, states which remote users are allowed to access the local data residing with peer *p* [24]. The drawback for this approach is that a centralized authority is required for user identification and peer identification [24]. As the network scales it becomes difficult to maintain the access control mapping because these mappings are stored by the peer responsible for the data.

The second approach is to use this XACML to create a Distributed Access Control Directory (DACD) [24]. This directory is used to store the access control information of the data in the network, which roles can access which data objects, which users can access which data objects. When a peer *p* gives access to any remote user to its local data or any particular user role the access to its data, this access control information is updated in the DACD. A local copy of this access information is also stored with peer *p* for perform and also as backup of the access informations [24]. Here in this approach its required that the peers and the users needs to be uniquely identified like in the XACML mapping approach [24]. The directory stored with the access control information is like a centralized server, where every peer must lookup the access right for the user to the requesting data [24]. Maintaining consistency of the DACD in a decentralized environment is huge problem because it should only allow a single access control record to be updated or modified at a given instance of time [24]. The directory itself should be protected from malicious peers in the network. Since there is no centralized authority in peer-to-peer networks, there should be restrictions on the access of the access control information records by the peers in the network [24]. Malicious peers should not be allowed modify or insert access control records of the data objects residing in other peers in the network.

Access control mechanisms can ensure the unauthorized data modification by any peer or user in the network. For this task, the peer and users in the network must be identified uniquely. Its better to use the DACD approach for large scale peer-to-peer networks and the XACML mapping approach for small scale peer-to-peer networks to protect the data against unauthorized modification or access [24]. Access controls on the data will definitely help in minimize the poisoning attacks, described in section 2.2.

4.7 Cryptree - Cryptographic approach

Cryptree [25] is tree based folder structure that uses the concept of access control mechanisms for guaranteeing security to the data in the peer-to-peer networks. Typically it's a tree based data structure. This data structure contains cryptographic keys and cryptographic link [25]. Combing the cryptographic links, a cryptographic data structure is formed [25]. The cryptree data structure is formed in this manner. Cryptree uses keys to perform read and write operations of the data.

A cryptographic link [25] can be described for two keys $C_{k_1} \rightarrow C_{k_2}$ as — any peer in that have the key C_{k_1} can derive the key C_{k_2} from C_{k_1} [25]. But the key C_{k_1} cannot be derive from C_{k_2} i.e. reverse derivation of the key is not possible unless there exists a cryptographic link by which C_{k_1} can be derived from C_{k_2}. There are two types of cryptographic links used

- Symmetric link — for symmetric cryptographic link between $C_{k_1} \rightarrow C_{k_2}$ is achieved by encrypting C_{k_2} with C_{k_1} [25]. Most of the cryptographic links except the cryptographic link between clearance key and subfolder key is symmetric.

- Asymmetric link — for asymmetric cryptographic link link between $C_{k_1} \rightharpoonup C_{k_2}$ is achieved by encrypting C_{k_2} with the C_{k_1} public key. Only with C_{k_1} private key knowledge C_{k_2} can be retrieved [25]. The cryptographic link between the clearance key and sub-folder key is always asymmetric for both read and write operations.

In read operation, cryptree uses 5 keys —

- Data key DK_f to encrypted data [25]
- Back link key BK_f to find information about the parent folders [25]
- Sub-folder key SK_f to read sub-folders [25]
- File key FK_f to read the files in the folder [25]
- Clearance key CK_f, optional key, given to other peers to read the file in the folders and sub-folders [25]

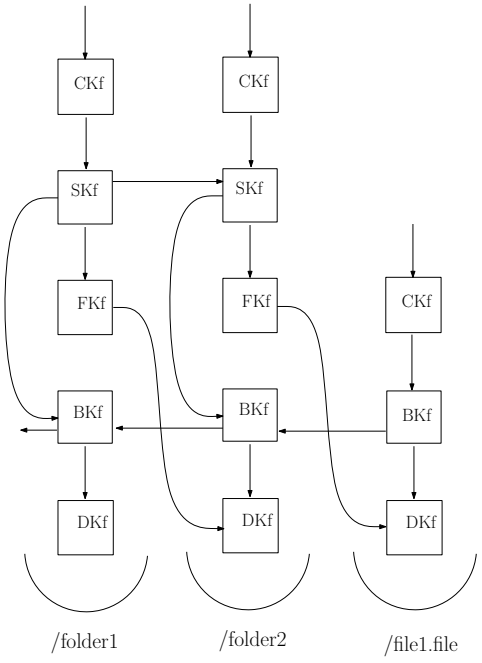


Figure 2. Cryptree cryptographic links used to generate the required keys for reading file1.file stored in folder2.

Source : Cryptree cryptographic links [25].

Cryptographic links are created between these keys to ensure security of the data. The links are created in the way which any user *u1* would want to grant access to folders or files to any other user in the network, explained in [25]. Figure 2 shows how the cryptographic links are used to generate the required cryptographic keys. In the figure 2, once the peer has the clearance key CK_f, it can use the cryptographic links to derive the Sub-folder key SK_f. Then the sub-folder key SK_f is used to derive the back link key BK_f and the file key FK_f, from which data key DK_f is derived. This data key DK_f is used to retrieve the file by decrypting the encrypted file. To grant access to the a particular file, clearance key CK_f and the back link key BK_f are linked to the other keys in the folder creating a cryptographic link so that the other peer cannot use the cryptographic links to derive the keys required to read the other files [25].

In write operation, cryptree uses the following additional keys alongside some of the keys in read operations —

- Verification key VK_f to verify the changes made during write operations [25]
- Signature key $SignK_f$ to sign the changes made during write operations [25]
- Write Sub-folder key WSK_f to provide write access for the sub-folders [25]
- Write Clearance key WCK_f , optional key, given to other peers to grant write access in the folders and sub-folders [25]

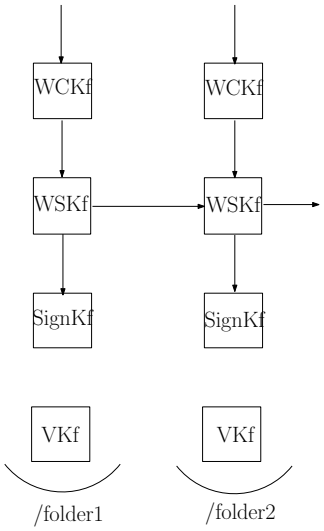


Figure 3. Cryptree cryptographic links used to generate the required keys for writing into folder2.

Source : Cryptree cryptographic links [25].

Figure 3 shows the cryptographic links for writing data. Once the peer gets the clearance key WCK_f, it can perform write operations on the particular folder and its sub-folder. By using the cryptographic links between the keys shown in

figure 3, the peer having the clearance key WCK_f can derive the write sub-folder key WSK_f. The write sub-folder key WSK_f is then used to derive the signature key $SignK_f$. The write operations are then signed using the signature key $SignK_f$. Verification key VK_f, which is publicly available is then used to verify the write operations done by the peers. The cryptographic links between the keys are created in such a manner that the peer cannot go back to the parent folder or write outside the folders it has permissions for [25].

Once the peer or the user has performed the read and write operations and verified their operations, their access to the data is revoked. This is done to prevent any modification or unauthorized access to data stored in the network. In this operations, the keys associated with the folders, the back link key BK_f and the clearance keys WCK_f and CK_f are cleaned or removed [25]. New and fresh keys are replaced so that the peer may not continue to access or write data in the network. In this method the users and the peers in the network should be uniquely identified. Cryptree [25] uses cryptographic links and keys to achieve the access control mechanisms [24], described in section 4.6. This prevents malicious peers or unauthorized users in accessing or modifying data in the network thereby protecting data integrity and minimizing the poisoning attacks, described in section 2.2.

4.8 Steganographic mechanism

Steganographic mechanisms are implemented in Mnemosyne [19]. The steganographic storage mechanism aims to hide the encrypted data among the other random data blocks inserted into the storage [19]. So any malicious peers or attacker has no way of identifying the presence of data in the storage until the malicious peer or the attackers gets hold of the cryptographic key used which encrypts the data. Once in possession of the key, the malicious peers can identify the random data blocks and the useful data in the storage.

In the steganographic approach, it's assumed that the storage can store *N* blocks of data. These *N* blocks are filled with random data [19]. The file is encrypted with the cryptographic key C_k, which is 256 bit AES key [19]. This encrypted file blocks are written into storage blocks at chosen location in the storage. This location is determined by an initial hash value h_0 and a validity check for verifying the data block [19]. The has function used is SHA-256 [19]. The data is stored by —

- Computing sequence of *n* hash values h_0, ..., h_{n-1} = $H(h_{n-2})$ [19]
- The replica *i* where ($0 \leq$ i $\leq$ n-1) is encrypted using the sub key $C_{k_i} = E_{C_k}(h_i)$ and stored at block $b_i = h_i$ mod *N* [19]

To read the data, the cryptographic key C_k and initial hash value h_0 is required [19]. Then each block b_i where ($0 \leq$ i $\leq$ n-1) is read, decrypted and the validity is checked until a valid block of data is encountered [19].

In peer-to-peer networks, to store the data we require the peer id and the block id on the peer to store the data. In steganographic method, we use $b_i = h_i$ mod *N* to store the data at block b_i [19]. From the 256 bit hash value h_i, we use the leading *k* bits used to identify the peers in the network by using the underlying peer-to-peer network routing schemes [19]. For example in Pastry [26] which uses 128 bits to identify peers in the network [26], the first 128 bits form the 256 bit hash value can be used to identify the peer in the network and of the rest remaining bits, example (32 bits), can be used to identify the blocks were the encrypted data is to be stored [19]. When a peer request the data, from the initial hash value h_0, using the first *k* bits peer id for the peer *p* is extracted and then blocks where the data is stored are read from the peer *p*, decrypted and the validity is checked until the first valid block of data is found [19].

Since all the data blocks in the storage are filled with either random data or the encrypted data, using steganographic methods provides anonymity in data stored in the network. Without the key C_k and initial hash value h_0 and validity check's knowledge, any malicious peers or attackers cannot modify or corrupt the data as the malicious peers cannot differentiate between the random data blocks and the encrypted data blocks. The malicious peers cannot write into the data into any random block because whatever data any peer in the network needs to write has to go through the write operation. Write operation has a validity check and the hash functions can guarantee with a good probability that modified data could be written into a different block. With quota management for peers, malicious peers cannot go on to write data into all the storage blocks available. This helps in minimizing the poisoning attacks described in section 2.2.

StegTorrent [27] is an example in which steganographic mechanism are implemented for communication over network. In StegTorrent [27], a steganographic mechanism for peer-to-peer file transfer service, is implemented to modify the order of data in the packets used for peer-to-peer data exchange and this modified packets does not require synchronization [27]. This technique is useful for distributing copyrighted contents maintaining anonymity while the contents are transfered [27]. Using this technique in communication can help to minimize the network pollution [3], described in section 2.2.1. The detail operation of StegTorrent in not in scope of this paper, it can be found in the paper *"StegTorrent: a Steganographic Method for the P2P File Sharing Service "* [27].

Steganographic mechanisms helps in securing the data and provide anonymity in the storage of the data and in

communication or transfer of data in peer-to-peer networks.

4.9 Trust and reputation mechanism

Trust based model and reputation mechanisms are also useful in enhancing the security in peer-to-peer networks. In this section we take a look in brief on some of the trust based security models and the reputation models that are available for implementation in the network. Trust and reputation mechanisms heps in minimizing the effect of poisoning attacks, described in section 2.2, on the data. It also helps in identification of the malicious peers in the network by using the trust value and thereby minimizing the effect of malicious peers on peer-to-peer networks.

4.9.1 Trust and reputation model

The trust and reputation model proposed uses the Bayesian network-based trust model[28]. Trust is defined as the belief of a peer in any other peer to provide correct and genuine data or service upon request from its own experience [28]. Reputation is defined as the belief of a peer in any other peer to provide correct and genuine data or service upon request from the other neighboring peers experience [28]. The trust and reputation model proposed has two types of trust between peers *p1* and *p2* —

- The trust *p1* has on *p2* to provide correct and genuine data [28]
- The trust *p1* has on *p2* to provide information on any other peers in network on their trustworthiness [28], the reputation of the other peers in the network.

The file or data sharing in the peer-to-peer network implementing Bayesian network-based trust model [28] has two scenarios. In first scenario, any requesting peer P_r contacts the trustworthy peer P_s that has the data based on its own previous experience [28]. The peer P_s will sent the requested data to the requesting peer P_r. In the second scenario, if the requested data is with peer P_s and the requesting peer P_r does not know about the trustworthiness of peer P_s, then the requesting peer P_r ask its neighbors or other peers in the network on the trustworthiness of peer P_s [28]. Now the requesting peer P_r contacts P_s for the data. In both the scenario, if the interaction was good and satisfactory, then the peer P_s goes up, else it goes down for both the scenario [28]. In second scenario in addition to the peer P_s trust value calculation, the requesting peer P_r will also update its trust values in the peers that gave recommendation on the peer P_s [28]. If the recommendation was correct with the trust values given by the other peers (recommender's) on peer P_s, the interaction was good and satisfactory, then trust value of the recommender's will go up else goes down [28].

4.9.2 Role based trust approach

The role based trust model proposed [29], assigns trust values to every peer in the network [29]. Higher the trust value, highly trustworthy is the peer in the network. The trust values assigned to peer depends on the relationship of the peer with the other peers in the network [29]. Any malicious peer that provides junk data or performs poisoning attacks, the requesting peer has the ability to drop the relationship with the malicious peer [29], i.e. the connection to the malicious peer is terminated. Once this happens, the trust value of the malicious peers comes down. By this method, any other peers in the network can identify which peers are malicious and which are trustworthy. Any peer that needs a particular data can connect to the trustworthy peer, i.e. form relationship to the trustworthy peer and request the data it required.

To calculate the trust value of peer in the peer-to-peer network or peer communities [29], peer links are used. A peer community is a group of peers that have common interests [29]. Peer links are defined as connections between the peers [29]. 1-hop links are direct connection from the peer, also referred to as its neighbors [29] and 2-hop links are the indirect connects to the peer in the network [29]. The peer links is used to calculate the link weights [29], the link weights is used to calculate the trust value of the peer in the peer communities and in peer-to-peer networks [29].

4.9.3 EigenTrust algorithm for reputation management

The EigenTrust algorithm [30] can be used to prevent the drawbacks in the previous trust and reputation models described. In the previous models, all the operations are based on trust values assigned to the peers in the network. The problems that can occur with this assignment is that the groups of malicious peers can assign each others very high reputation values. When the other good peers uses this values to retrieve data, the malicious peers can present junk data, causing poisoned attacks, described in section 2.2. In addition the group of malicious peers can cause routing attacks, described in section 2.3. This will lead to the storage and retrieval attacks [2] described in section 2.1.

To prevent these issues, the EigenTrust algorithm [30] uses these local trust values to calculate the global reputation value of the peers in the network [30]. The global reputation values are calculated by using the local trust values assigned to peer *p* by other peers, weighted by the global trust values of the assigning peers [30]. The algorithm for calculation of the global trust value in peer-to-peer network is known as the Distributed EigenTrust Algorithm [30]. The Distributed EigenTrust Algorithm is described in detail in the paper *"The EigenTrust Algorithm for Reputation Management in P2P Networks "* [30].

To ensure that the malicious peers in the network cannot influence the calculation of a peers global trust values, multiple peers calculate the trust value of any other peer in the network, called score managers [30]. This algorithm is also known as Secure EigenTrust [30]. The score managers are assigned by using hash functions on the peer *p's* id and these hash values are mapped into DHT (Distributed Hash Tables) indexes [30]. Any peer on these mapped index of the DHT becomes the score managers for the peer *p* [30]. When any requesting peer in the network need to know any other peer *p's* trust value, the requesting peer can ask the score managers of the peer *p* to get the trust values [30]. In case of conflict of the trust values, the value in which the majority of the score managers agrees on, is chosen as the trust value of the peer in the network [30]. This helps in minimizing the false trust scores presented by the malicious peers in the network for any other peer in the network and minimize the effect of malicious peers in the network.

4.9.4 Fuzzy trust approach

In this trust and reputation based model, fuzzy logic is implemented to manage trust and reputation [31] of the peers in the network. Fuzzy logic is used because it provides the ability to handle uncertainty. Fuzzy logic can then map this uncertainty in trust values of peers into fuzzy sets with a value between [*x,y*] (where $x,y \in Z$ (set of integers)), *x* can represent no membership and *y* can represent complete membership to the fuzzy sets based on the fuzzy rules [31].

The fuzzy trust model proposed [31], the trust depends on the peers individual experience of interaction with the other peers in the network [31]. Form this interaction history of peers in the network, the trust values are calculated. The trust is modeled into different dimension in the system [31], which can be factors like latency, authenticity of the data etc... To calculate the trust of a peer *p*, The interacting peer *q*, calculates the experience of each dimensions *d*, of trust defined in the system [31]. The experience is calculated using the formula

$$E_p^d = \frac{I_p^{d+} - I_p^{d-}}{I_p^{d+} + I_p^{d-}} [31] \quad (1)$$

where I_p^{d+} value corresponding to the number of interaction in which peer *p* met *q's* expectation and I_p^{d-} value corresponding to the number of interaction in which peer *p* didn't met *q's* expectation [31]. This E_p^d is transformed to fuzzy value sing the fuzzy sets [31]. The peer *q* should have sufficient interactions for calculation of the trust value for a dimension *d*. If not a default trust value is used [31].

The value of E_p^d, the experience lies in the range [-1,1] [31]. These range values are classified into — negative big, negative medium, negative small, zero, positive small, positive medium, and positive big [31]. -1 stands for negative big experience and +1 stands for positive big experience [31]. The trust value for peer *p* is defined as T_p, a fuzzy variable whose value lie in the range [-1,1] [31]. This range of values is further classified into fuzzy sets — high distrust, distrust, undistrust, untrust, trust, and high trust, where -1 represents high distrust and +1 high trust [31].

Fuzzy rules are defined, explained in detail in the paper *"Fuzzy Trust for Peer-to-Peer Systems "* [31]. Using these fuzzy rules, the trust values T_p for a peer *p* is calculated from the experience values of each trust dimension E_p^d calculated using equation (1) by other peers in the network for the peer *p* [31]. This trust value T_p and the fuzzy set for trust values is used to determine the trustworthiness of the peer *p* in the network.

5 Conclusion

We have seen some of the attacks that can occur in the data and the storage in peer-to-peer networks. To prevent these attacks, identification of malicious peers must be done effectively.

The replication algorithm employed should ensure data availability in the network and avoid single point of responsibility for the data in the network. The replication algorithm should ensure the data is replicated in the network correctly and enough replicas should be maintained so that the latency in data access is minimized and the load in the network is distributed. In my opinion it's recommended to implement the introspective replica management alongside the replication strategy used in PAST. The peer-to-peer framework which uses PAST replication scheme and cache-based replication can also incorporate introspection techniques by using value recordings obtained from systems monitoring functionality implemented in the present project group. Table 1 shows the conclusion of the replication algorithm.

Quota management must be employed to stop the malicious peers to inject large amount of junk data that can cause poisoning attacks in the network. This would be managed by smartcards used in PAST that helps to identify the peers in the network. To protect the data, data can be encrypted with any cryptographic encryption schemes. The key management of this encryption scheme implemented should be taken care so that any malicious peers present in the network cannot damage the data. The keys used for encryption can be stored and retrieved using secure key management schemes. Shamir's secret sharing scheme, Information dispersal algorithm or erasure codes can be used to manage the cryptographic keys used for encryption of the data.

Integrity of the data can be ensured with the help of cryptographic hash functions and verifying them every time any data is downloaded by any peer. The data should be protected using the security mechanisms or by using the combination of these security mechanisms to form sophisticated security mechanisms for the data in the network. Access

Replication	Recommended	Reason
Passive	✓	Can be used alongside PAST replication algorithm. The destination peer can then use this file to serve the request in its locality. Modification should be made to incorporate the data update in the network.
Active	X	PAST replication takes care of maintaining enough replicas in the network. Using this algorithm could flood the network with data. Too many replicas would be available when the network doesn't need that many replicas to be available.
Cache-based	✓	Already implemented in PAST.
Dynamic	X	This algorithm would help in maintaining the required amount of replicas in the network and provides better quality of service. This technique can be considered, but might be complicated to implement.
Introspective	✓	Algorithm can be used to maintain fair amount replicas by using the system monitoring functionality to reduce the latency in file access.

Table 1: Summarization of replication algorithms

Security mechanism	Recommended	Reason
Self certifying data	✓	Available with the smartcards used in PAST
Information dispersal algorithm	X	It's better to use erasure codes, since it's use the same concept and is much more reliable on untrusted environment.
Shamir's secret sharing scheme	X	It's better to use erasure codes. Information dispersal algorithm is the enhancement of Shamir's secret sharing scheme.
Smartcards	✓	Already available in PAST for user identification. It also can be used for quota management and verifying data.
Erasure codes	✓	Secure decentralized erasure codes can be considered for implementation. The Asymmetric cryptographic key used for homomorphic cipher text must be managed by using erasure codes, Information dispersal algorithm, Shamir's secret sharing scheme or any other key management schemes.
Access control	X	Access control is necessary for restricting unauthorized access to the data. Access control list should be maintained along with the data instead of the peer responsible for the data storing a complicated list of users. The approach is a bit complicated to implement and maintain.
Cryptree	✓	An alternative approach to the access control mechanism. Cryptographic keys ensure more security and it provides access control mechanisms for the data or folders in the network.
Steganography	X	There exists a high probability for hash collisions to occur while storing data. To avoid this multiple hash functions are used, but still the data could be completely lost in the network due to hash collisions.
Trust and reputation	✓	The fuzzy trust calculation mechanism can be combined with EigenTrust algorithm so that there is an acceptable trust value range. The fuzzy logic helps to handle the uncertainty in trust values.

Table 2: Summarization of security mechanisms

control mechanisms should be incorporated so that the data is protected from unauthorized access. The ACL (Access Control List) can be maintained alongside data. In my opinion Cryptree can be useful for implementing the access control mechanism by using cryptographic keys. I would also recommended to use trust and reputation mechanisms can be employed so that the peers know which are very good sources to get the data and which peers can present useless and junk information. The problem with malicious peers pushing up their trust value can be solved by using fuzzy trust and secure EigenTrust algorithm. Table 2 shows the conclusion of the security mechanisms.

References

[1] Y. Zhu and Y. Hu, "Routing, storage management and caching, and security of peer-to-peer storage systems," in *Department of Electrical and Computer Engineering and Computer Science, University of Cincinnati*.

[2] E. Sit and R. Morris, "Security considerations for peer-to-peer distributed hash tables," in *Peer-to-Peer Systems. Springer Berlin Heidelberg*, 2002.

[3] N. Christin, A. S. Weigend, and J. Chuang, "Content availability, pollution and poisoning in file sharing peer-to-peer networks," in *Proceedings of the ACM conference on Electronic commerce*, 2005.

[4] J. Liang, N. Naoumov, and K. W. Ross, "The index poisoning attack in p2p file sharing systems," in *IEEE INFOCOM*, 2006.

[5] J. R. Douceur, "The sybil attack," in *Peer-to-Peer Systems. Springer Berlin Heidelberg*, 2002.

[6] A. Singh, T.-W. . Ngan, P. Druschel, and D. S. Wallach, "Eclipse attacks on overlay networks: Threats and defenses," in *IEEE INFOCOM*, 2006.

[7] A. Kaluszka, "Distributed hash tables," in *University of California, Berkeley*, 2010.

[8] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," in *ACM Computing Surveys (CSUR)*, 2004.

[9] E. Sit, A. Haeberlen, F. Dabek, B. Chun, H. Weatherspoon, R. Morris, M. F. Kaashoek, and J. Kubiatowicz, "Proactive replication for data durability," in *International Workshop on Peer-to-Peer Systems (IPTPS)*, 2006.

[10] P. Druschel and A. Rowstron, "Past: A large-scale, persistent peer-to-peer storage utility.," in *Proceedings of the IEEE workshop on Hot Topics in Operating Systems*, 2001.

[11] P. Druschel and A. Rowstron, "Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility," in *ACM SIGOPS Operating Systems Review*, 2001.

[12] M. Waldvogel, P. Hurley, and D. Bauer, "Dynamic replica management in distributed hash tables," in *Bibliothek der Universitt Konstanz*, 2003.

[13] J. Kubiatowicz, D. Bindel, Y. Chen, P. E. Steven Czerwinski, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "Oceanstore: An architecture for global-scale persistent storage," in *ACM Sigplan Notices*, 2000.

[14] H. Weatherspoon, Y. Chen, and D. Geels, "Introspective replica management," in *University of California, Berkeley*.

[15] V. Pless, "Introduction to the theory of error-correcting codes," in *Wiley Interscience*, 2011.

[16] D. S. Wallach, "A survey of peer-to-peer security issues," in *Software Security - Theories and Systems. Springer Berlin Heidelberg*, 2003.

[17] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with cfs," in *ACM SIGOPS Operating Systems*, 2001.

[18] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," in *Journal of the ACM (JACM)*, 1989.

[19] S. Hand and T. Roscoe, "Mnemosyne: Peer-to-peer steganographic storage," in *Peer-to-Peer Systems. Springer Berlin Heidelberg*, 2002.

[20] A. Shamir, "How to share a secret," in *Communications of the ACM*, 1979.

[21] H.-Y. Lin and W.-G. Tzeng, "A secure decentralized erasure code for distributed networked storage," in *IEEE Transactions on Parallel and Distributed Systems*, 2010.

[22] Z. Zhang and Q. Lian, "Reperasure: Replication protocol using erasure-code in peer-to-peer storage network," in *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, 2004.

[23] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," in *IEEE/ACM Transactions on Networking (TON)*, 2006.

[24] C. Sturm, K. R. Dittrich, and P. Ziegler, "An access control mechanism for p2p collaborations," in *Proceedings of the international workshop on Data management in peer-to-peer systems*, 2008.

[25] D. Grolimund, L. Meisser, S. Schmid, and R. Wattenhofer, "Cryptree: A folder tree structure for cryptographic file systems," in *IEEE Symposium on Reliable Distributed Systems, SRDS'06*, 2006.

[26] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," in *Middleware. Springer Berlin Heidelberg*, 2001.

[27] P. Kopiczko, W. Mazurczyk, and K. Szczypiorski, "Stegtorrent: a steganographic method for the p2p file sharing service," in *arXiv preprint*, 2013.

[28] Y. Wang and J. Vassileva, "Trust and reputation model in peer-to-peer networks," in *Proceedings of the IEEE International Conference on Peer-to-Peer Computing*, 2003.

[29] M. Khambatti, P. Dasgupta, and K. D. Ryu, "A role-based trust model for peer-to-peer communities and dynamic coalitions," in *Proceedings of the IEEE international Conference on Information assurance workshop*, 2004.

[30] S. D. Kamvar, M. T. Schlosser, and H. GarciaMolina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the ACM international conference on World Wide Web*, 2003.

[31] N. Griffiths, K.-M. Chao, and M. Younas, "Fuzzy trust for peer-to-peer systems," in *IEEE International Conference on Distributed Computing Systems Workshops, ICDCS Workshops*, 2006.

Secure Communication in Oppressed Countries

Maitri Modasia
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email: maitri.modasia2@gmail.com

Abstract

Decentralized Online Social Networks(DOSNs) are getting popular day by day as compared to other centralized server based networking sites like facebook. The reason for the gain of such high importance is the idea of distributed storage. Since the data is distributed in the network by a purely unbiased algorithm, none of the users are liable for the content technically. However, in server based systems, all the data is stored at a single place and hence there might be a threat to security or privacy of the data. In oppressing countries like China and many more, liability of users is blocked or they are not permitted access to certain data available online. This paper discusses in brief about the problems faced by users of such oppressing countries. Furthermore, we argue why Peer-to-Peer(P2P) technology plays an important role in such places and provide a brief overview on the existing solutions to avoid such problems.

1 Introduction

The world is getting smaller and smaller day by day due to the immense popularity of online social networking. According to former statistics, it has been seen that more than two-third of world population participates in social networking either for staying connected socially or professionally [18]. It is very much evident that the use of existing social networking websites is inevitable as they provide a lot of interesting functionalities as well as lot of facilities to explore and connect to the different users globally. But still their scope is limited when it comes to providing access rights to the users. Users refrain to adapt any new technology when they are quite adjusted with the existing and powerful social networking techniques. However impact of both Centralized and Decentralized OSNs have to be realized. In further sub-sections, we will discuss briefly about centralized and distributed stogare. Both the techniques have their own pros and cons. The trend of social networking is changing by adapting the idea of decentralization. The main idea is to realize the better technique out of the two approaches and focus more on the emerging P2P technology.

1.1 Centralized vs Distributed Storage

Centralized storage: It refers to the traditional Client-Server Architecture in which there is one server that acts as a higher performance system and the only provider of the content and service. The clients (PCs, workstations etc.), on which the users run their applications, request or execute the services provided by the central server and do not share any of their own resources. Thus, it can be easily predicted that the clients rely completely on servers for any resources like files, devices, or even the processing power. They only do the work of sending user input and get the display of requested output [28]. Since we are talking under the pretext of online social networking, it is very important to consider the factor of data storage of users who are using such websites for online social interactions. Majority of service providers of social networking websites use this approach where the user data is stored on a centralized main server owned by them. However, there are advantages as well as disadvantages which are discussed below in brief [28].

Advantages:

- Since all the data or files are stored at one location, its management becomes a lot easier.
- All the user data at the server is managed by the system administrators authorized by the service providers which prevents the users from manipulating any data.
- Using a central server for data storage helps in managing and reducing the network traffic.

Disadvantages:

- Centralized servers require a lot of maintenance and are expensive.
- Users do not have any track or control over their own data. They are not granted access rights on their own data. Moreover, there has always been a threat regarding privacy and security concerns.

- There is always a high risk of single point of failure which results into complete break-down of the whole network affecting the clients working on it.
- There is a great loss to the business processes that totally rely on such data storage approach.

Distributed Storage: Although, the centralized online social networks are extremely popular, we still see many issues regarding the privacy concerns and availability of service to the users. Thus, efforts have been made to overcome these problems by introducing the idea of P2P technology which is based on P2P architecture. The data or content is shared among the end-users or peers, thus implementing the decentralized approach unlike Client-Server architecture. Peer to Peer applications are well known for file-sharing, audio and video streaming. However, P2P framework for social networks has been successfully developed in recent years. Decentralized data storage has its own advantages and disadvantages too [28].

Advantages:

- Unlike the centralized storage systems, copies or replica of data is available on different peer systems resulting into the higher availability of data.
- Each peer shares the load of processing power of computations, storage capacity and cost which in turn provides the better utilization of unused resources of other peers.
- P2P networks are highly resilient to DoS attacks and network faults. Such technology minimizes the risk of single point of failure.
- The users have the control of their own data and along with appropriate encryption techniques; they can decide whom to give access to the data.
- It results into low bandwidth consumption comparatively.

Disadvantages:

- P2P networks are also vulnerable to threats like malicious attacks if any of the peer is sharing harmful data.
- When individual peers are offline, data is unavailable.
- Searching of content is easier from a centralized database rather than using metadata search of peers. However, such problems can be overcome using hybrid P2P techniques.

1.2 Changing Trend in Social Networking

From the previous sub-section, a complete overview on current trends in online social networking can be obtained. Decentralization is considered as a stepping stone in the history of social networking. Several Online Social Networks(OSNs) like Facebook, Twitter, Xing etc., based on centralized server, has two problems:1.The information on one website is not usable on others. 2.Privacy problems. The representation of user data such as photos, personal wall, messages, updates and user posts is exclusively managed according to the owner of the website. Even though, the user deactivates his account, strands of his personal information still prevails on the server. It is very difficult to transfer the user data to another social networking site, which also, is a very tedious task. Portability of own data is not possible. Considering these problems, P2P networks adapt a framework which works on different alternative technologies. P2P applications started emerging as a result of need for distributed storage of data and to overcome the monopoly of service providers. Early P2P networks were specifically used for file sharing. First and second generation P2P applications like Napster [31] and Gnuttela [20] respectively became popular for this reason. Napster, a hybrid P2P network, is susceptible to single point of failure, as it uses a centralized indexing server for lookup. Similarly, Gnutella and Kazaa [26] which implements a totally distributed approach without a central indexing server can have possibilities of non scalability and unresponsiveness, if the peers sharing the content are not in the network [5]. During the passage of time, more and more complex and sophisticated approaches of P2P networking emerged such as P2P overlay networks and are still open to research and implementation. These P2P applications which are better off than their predecessors provides more privacy,data security, resilience to monopoly, censorship and other issues. With such DOSNs, users can interact with other users beyond the boundaries of social networking websites giving them more freedom to get connected [37].

1.3 Organization

The idea of writing this paper is to discuss the internet censorship in oppressed countries and the problems faced by the users due to it. Section 2 talks about different techniques used by the governement of oppressed countries to control the internet. It also shows some examples, where people tried to use internet as a medium for sharing information with the outside world and had to face its drastic consequences. Section 3 lists some of the decentralized approaches which can result into a boon for social networking for such oppressed countries. Section 4 provides a conclusion by advocating the implementation of P2P techniques for social networks.

2 Online Communication In Oppressed Countries

Freedom of speech, expression of ones thoughts and ideas is a fundamental right of all human beings. Citizens of democratic republic countries enjoy the liberty of expressing their views. Moreover, in today's time, OSNs and other internet technologies like blogging, content sharing etc., also play a pivotal role of social media, which can be used to publish, express or upload any content online and share the information of any social or political issues. This has become extremely useful in establishing reformation movements for betterment of people. Sensitive social issues, inhuman activities, political oppressions, etc. have been exposed to the world via internet. Such is the power of "Social Internet" because of which any kind of news reaches each and every corner of the world. But, in oppressing countries like China, Iran, Tunisia, North Korea, Egypt, Syria and many others, the power of people is limited. In this section, the problems faced by the people of such oppressed countries, role of authoritarian government and the alternative solutions pertaining to those problems are discussed on a brief note.

2.1 Internet Censorship In Oppressed Countries

Unlike Democratic countries, where the citizens are given rights of accessing and using internet with full freedom, internet censorship is a questionable topic for oppressed countries. The government of such countries is autonomous by nature and limits the freedom of its citizens in many ways, including internet access. In order to suppress the people and to prevent the political opposition, government also controls the flow of data over internet. Social networking websites like Facebook, Twitter etc. are blocked in countries like China and many others, in order to restrain people from sharing the content and socializing globally [32]. Moreover, the online activities are constantly under surveillance and there is a high possibility of getting tracked. If anyone is caught accessing the blocked content, any kind of legal actions could be taken against them. Hence, in such conditions, internet access might jeopardise its users [22].

2.2 Role of Government, Internet Service Providers and Users

Since the oppressed countries have an authoritarian government which holds a supreme power, it is very easy to constantly monitor the actions of its people. There are several reasons behind it. First of all, the political hold over people needs to be maintained to sustain with such autonomy. Secondly, the government does not want to disclose any sensitive issues or movements to the outside world which could be possible through sharing any such manifestos, videos, articles, discussions or images online. Lastly, undesired content like pornography, that can adversely affect the citizens needs to be blocked. Note that such type of government not only has power over traditional media, police force or political bodies but also, over the internet. Moreover, the Internet Service Providers(ISPs) and communication companies can be easily controlled in countries where they operate. Some of the best examples of practices of internet censorship in oppressing countries are listed below:

1. Iran is one of the oppressed countries where internet censorship is practiced religiously. In June 2009, during the presidential election, the images of protests were sent via internet by ordinary citizens to the outside world [25]. Also, well-known Iran Green Movement was carried out in 2011, to spread the revolutionary ideas and gather worldwide support for the oppressed people [24]. Unfortunately, due to high surveillance and monitoring over the internet, people such online activists were severely punished and imprisoned by the Government. Moreover, all the green movement websites were hacked.

2. The Egyptian's Internet Shutdown [36] and "Arab Spring Revolution" [33] instances are popular worldwide which describe the activities of oppressive government against their own people.

3. Few countries like Burma, Cuba, North Korea and Turkmenistan provide internet access to limited number of people and also the cost of internet is high, so most of the citizens of such countries cant afford to use it [25].

4. Countries like China, Iran and Tunisia have highly efficient, well-organized multilayered systems for censoring the OSNs. The great firewall of China [35]is an excellent example of internet censorship.

5. During the Libyan Civil War'2011 and "Syrian Uprising" [25], government was able to identify and track down the protestors and online activists through their powerful internet surveillance which resulted into strict punishment.

   Thus, from the examples given above, it can be seen that there are several ways through which the government can restrict the access to internet like shutting down the internet completely, shunning mobile services, not allowing the broadband at home, limiting the internet browsing, uploading and downloading speed, offering connections with low bandwidths, adopting different internet censorship methods like IP based, DNS based, Keyword-based [17] etc. However, outsourcing of such services are sometimes, given to private communication companies, ISPs and blog hosting

companies. If such companies fail to fulfill the government demands, then their licenses could be taken away. Also, in some places, the users are required to register with an ISP when they want to purchase the internet and the cybercafes have to maintain the record of 30 days browsing history of the customers. Evgeny Morozov known for coining the terms such as Digital Dictatorship, has twitted that, Internet may actually be more useful to Authoritarian Regimes than to popular movement that oppose them [25].

2.3 Need for Solution to Existing Problems

It is rightly said that "If we have a problem, there exists an obvious solution". Technology has always stepped forward to overcome the loopholes in the existing system. OSNs like Facebook, Twitter etc., are based on a centralized server system which can be easily monitored and tracked. Thus, there was a need of introducing distributed or decentralized approach for social networking. P2P networks are alternative solution to the existing problems in oppressed countries. The use of proxy servers, alternate DNS root name servers, private alternative network devices, encrypted & distributed communication and anonymization techniques are some of the approaches of P2P networking. Many softwares and applications like Tor [13], Psiphon [7], FreedomBox [8], Diaspora [14] etc., have successfully worked out positively so that sharing of data online, without a threat of being identified or tracked, is possible. This idea works as no data is registered at one place and all participants are anonymously distributed over a network. These P2P approaches are discussed in detail in the further section of this paper. Current OSNs might be considered as highly advanced applications but, their technical deficiencies and limitations can result into great problems as we have discussed in this section. Moreover, these problems have created public awareness and we are heading towards more sophisticated and secured web technologies. Hillary Clinton has stated in one of her speech, Spreading of technology will also result in spreading of democracy, freedom and human rights [25]. With the advancement in decentralization approach, we can expect to have secure communication in oppressed countries and preserve individual rights of free expression and speech.

3 Solutions

This section can be considered another major topic of discussion that will focus on different P2P approaches for decentralized online social-networking. We classify P2P applications according to the technical approach they follow. Furthermore, it is important to figure out a comparatively better approach which serves the purpose of security, anonymity as well as privacy of the users. We have many P2P applications which are based on different unbiased algorithms and mechanisms. Some of them are based on pure P2P networks whereas others are hybrid P2P applications. In the following sub-sections some popular applications along with the type of technical approach they implement for social networking or file sharing are discussed. Lastly, short comparison is done for general analysis of such applications when put into practice.

3.1 Own Server

One amongst many approaches of decentralization is to run a social network using own server. Its can be a group-based mechanism or a collaborative effort of the users of such groups to establish a collaborative social network which does not have a centralized server approach. The main goal of such an application is to link the members of a group to other communities or groups rather than connecting with individuals. Such a distributed system offers high security by preserving privacy of content shared among the anonymous peers as well as no threat from unreliable service providers. The peers have all the rights of deciding with whom they want to share their profile information, any files, giving other access rights etc. Some examples implementing this mechanism are given below:

Crabgrass: Crabgrass [9] is a web application which provides its users a platform to carry out the social networking activities like highly secured chatting, file sharing, bookmarking, democratic decision making, network organizing, rating content etc. Moreover, it helps the groups to create and diversify the network by linking other independent groups. It provides facilities like having a customized homepage, private messing, participating in forums and blogs, searching of files and many more. However, it promises high data security as the content is shared only between its members and not the outside world [6].

Buddypress: Buddypress [2] is a plugin installed on Wordpress which transforms it into a social network platform. Different communities or groups such as companies, educational institutes, sport organizations etc can build their own social networks and customize them by adding different GUI features to it such as additional plugins, menu buttons, themes, etc.[1]. Such application also does not require a centralized approach to store user data, thereby offering a secure communication.

3.2 Web Servers Via Standard Protocols

P2P networks also work on the idea of establishing private web-servers via standard protocols based on semantic web technology. One such protocol is FOAF (Friend-Of-A-Friend) [10]which uses RDF (Resource Description Framework) and OWL (Web Ontology Language) which

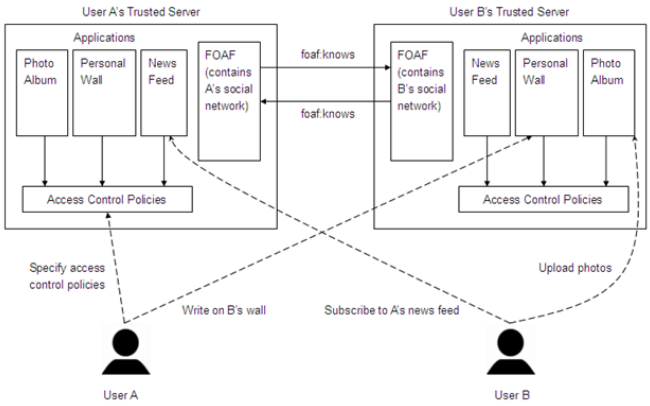


Figure 1. FOAF [25]

describes persons, their activities and their relations to other people and objects [3]. FOAF can be described as a vocabulary or a machine readable ontology which uses a decentralized approach for social networking. Users can create their FOAF profiles and can store them on trustworthy servers. FOAF works on standard web-based protocols such as WebDev and SPARQL/Update. Users can store their information at different places and can be connected to each other by their distinct WebIDs. These WebIDs are used as access point to get the FOAF file of the user and in turn get connected to the users social network [25]. Some of the examples based on this approach are:

Diaspora: Diaspora [14] is a free personal web-server which provides a distributed social network platform, through nodes called "pods". Users can set up their own server, host the content and share their own data.

FreedomBox: Another similar approach like Diaspora which promotes a personal web-server for distributed social networking, emailing, audio/video communication is FreedomBox [13]. "FreedomBoxes" can be termed as decentralized deployment of hardware to individual homes or offices, that will provide privacy, safe and secure communication for people of oppressive regimes.

HelloWorld Network: HelloWorld Network [12] is a client-software that runs on a users local machine. The users information on this social networking software is highly encrypted. It uses a HelloWorldID as an authentication protocol. There is a URI which points to an HTML page that has a metalink to users RSA public key. HelloWorld Network provides features like authentification, identification, decentralization and secure communication [15].

Other similar approaches include Fether (twitter) which provides the facility of micro-blogging and Diki which provides a decentralized, highly secured and privacy preserving bookmarking service. Diki is based on protocols like XMPP, a secure communication protocol [37].

3.3 Centralized Data Indexing

Internet searching through centralized search engines has become more complex due to the increasing number of websites and information on the web. Moreover, such central server based search engines are incapable of interpreting queries individually. P2P networks can be categorized as pure and hybrid networks. In hybrid P2P networks, we have a combination of peer to peer and client-server architecture. Such systems consist of a centralized server as a directory for look up and fetch peers address. Indexing helps to find whether the peer hosting the requested file is online and connected to the central server. Once the requested file is found, the server will tell the user about address of the requested file. After this, the user establishes the connection with the host computer and on successful connection, downloading of file is possible. Thus, it implements an idea of decentralization with centralized indexing for better search and promises a good scalability and security. Some of the P2P applications based on central data indexing are:

Maze: Maze [23] is a hybrid P2P system in which peers directly communicate to each other and list the data they want to host on a centralized server. However, the protocols used for communication and look up search are highly cryptographic in nature. Maze uses a Kerberos protocol for user authentication. Since some peers also use dynamic IP addresses, inorder to use Maze, a user needs to register with MazeID acquired through TGS(Ticket Grant Server) with personal password. Maze provides peer discovery in social P2P, file sharing and download, sharing friends list and addresses etc.

Gossple: Gossple provides a personalized web navigation, decentralized social network platform where user affinities and preferences is the heart of the search process [19]. Unlike, traditional search engines like Google, Gossple seeks the information directly from the user. Users with similar interests come across and link with each other anonymously and enhance their search operations [4]. It implements the gossip based algorithms that are highly efficient for dynamic indexing, searching, asynchronously transmitting the information to the users based on their preferences.

COBs: COBs is also a hybrid P2P network approach which is similar to Wikipedia. Internet browsing can be sometimes a tedious task if we do not find the relevant content online even after searching with keywords. It can work as a front-end, like a browser add-on that helps the users to

socialize with others in both asynchronous and synchronous manner. It also provides a decentralized back-end which can aim at a more sophisticated system with personalization and privacy [34].

3.4 Solutions Based on Existing Overlays

Decentralized social networking can be done using an emerging technology based on P2P overlay networks. Peers form an overlay over the communication network layer. An overlay is a self-organizing logical network with a single routing layer between the peers. P2P overlay networks are divided into two types: structured overlay and unstructured overlay. Structured P2P overlay is based on DHT(Distributed Hash Table) and uses it as a substrate. Peers contain information about NodeIDs and IP addresses of other peers. Data objects routing in the network are assigned unique identifiers called keys. The look up possibility for data in such DHT based systems can be very high and thus provide a wide, scalable retrieval of shared information. In Unstructured P2P overlay networks, the search is done by keywords instead of unique IDs. There are no specific algorithms or layout to find the files. There is no concept of special identifiers for data objects. All the peers are responsible for their own content submitted. P2P overlay networks function on trusted servers and provide higher characteristics like looking up resources, routing messages and effective communication among nodes in an overlay. Some of the examples of applications based on overlay are:

Kademlia: Kademlia [29] is a protocol for P2P networks which is based on DHT. In Kademlia, a virtual overlay network is formed by the parcicipant nodes and each node is identified by a nodeID. Look-up procedure in Kademlia becomes very efficient as nodeID provides direct mapping to file hashes from where the node source has to be obtained. Kademlia uses an XOR metric for distance calculation between the two nodes and a single routing algorithm to locate the nodes in the network. For joining the network, a peer or node must go through the bootstrap process, in which, it needs to know the IP address and port of another node(bootstrap node)that is already in Kademia network. The algorithm uses some protocol messages like PING, STORE, FINDNODE, FINDVALUE etc. This algorithm is resistant to denial of service attacks and thus becomes a good choice for decentralized social network. Kademlia is the first P2P system which minimizes the routing, provides faster lookup and avoids timeout delay in case of node failures. Safebook implements kademlia protocol for lookup.

Likir: Likir(Layer Identity based Kademlia like Infrastructure) [16] is an extension of Kademlia protocol which help in building up a secured P2P overlay network. P2P overlay networks are prone to different kinds of threats like sybil and denial of service attacks. Likir provides a strong node-identity service with its encrypted DHT system. All nodes in Likir consists of an identifier in the form of an OpenID by certification service [25]. Thus it provides a secure communication channel through node authentification and message integrity. Also, it enhances the possibility of data confidentiality and anonymity in the overlay network. Likir also provides a simple API, efficient routing, safe identity based data retrieval and reliability which creates an ideal framework for P2P social networking systems.

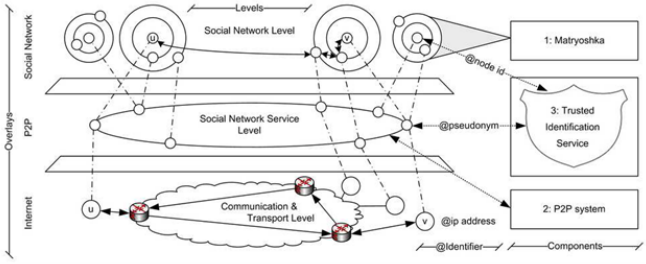


Figure 2. Safebook [25]

Safebook: Safebook is a distributed social networking application which is based on structured P2P overlay. Safebook implements similar structure as Kademlia. Safebook was mainly developed to provide security, integrity and data availability in a distributed manner. It consists of three main components namely TIS (Trusted Identity Service), matryoshkas and a P2P location substrate. TIS is a stateless, offline serve and perform a cryptographic function to issue keys. It does not store any information and is meant for identification and certification of users. Matryoshkas are specialized overlays that hide user information, increase user availability and replicate user profiles amongst different devices [25]. The figure shown here, presents the components and communication done in Safebook.

LifeSocial: LifeSocial [27] is plugin based architecture for P2P based distributed network which provides a secure and extensible functionality for online social networking. It is a structured P2P overlay which provides IT-based routing among the peers. There are specific UserIDs and cryptographic key to identify the users. LifeSocial is based on FreePastry [11] and PAST [30] techniques for data storage and data replication respectively. It provides a user-friendly GUI framework which consists of similar features as in popular OSNs along with additional features like common whiteboard, play online games etc. LifeSocial can be extended with various additional plugins easily. Technically,

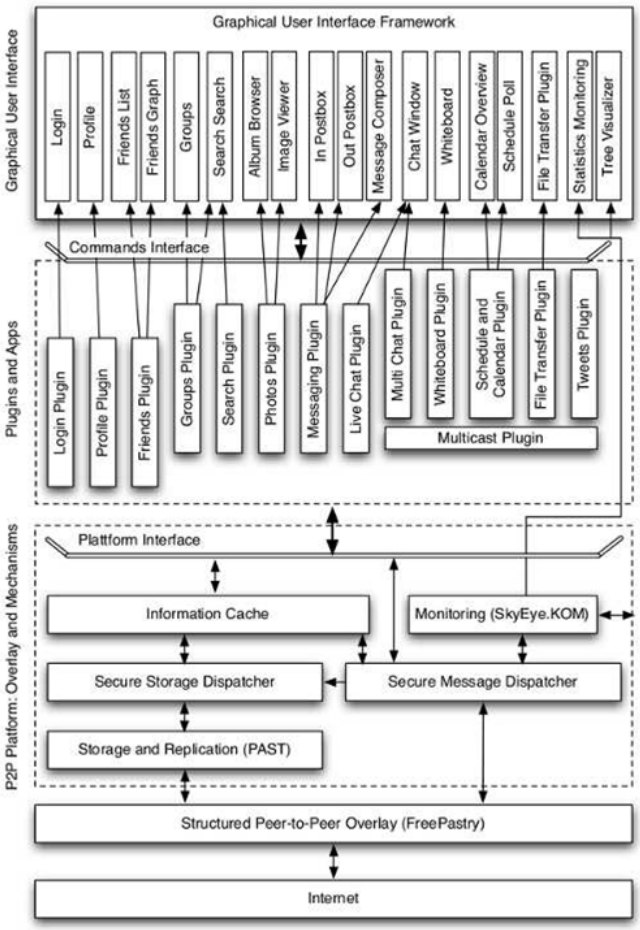


Figure 3. LifeSocial [25]

it provides high data availability due to replication of every document, consists of security layer for authentication, data communication etc. Its infrastructure is operated by the users and is quite feasible as the data load is distributed among peers. Such approach is cost-effective and requires low bandwidth. Figure given, shows an overview of plugin architecture of LifeSocial.

PeersON: PeersON [25] maintains the characteristics of OSNs by providing encryption, data access control decentralization and direct data exchange. Encryption ensures the privacy to the users, decentralized P2P infrastructure provides independence from the OSN service providers. Direct data exchange is possible without constant internet connectivity. PeersON allows asynchronous communication when peers are not online. It uses a DHT as lookup system, so the peers can connect directly and also provides digital personal spaces for users to maintain their own data.

3.5 Comparison

So far, we have seen different techniques implemented for decentralized social networking. Applications working on own servers provides a good collaborative social network but could have challenges when they are to be implemented on a large scale. P2P networks based on private servers via standard protocols have proved to be successful in terms of granting privacy and ownership of trusted servers. They create a social networking platform which is immune to internet censorship. But, the major challenge for such application is adoption by the users since they are already familier with current social networks [37]. Introducing interative GUIs, facilities of importing and exporting data, easy installations of softwares can be a good motivation to make such applications usable by the users. Lastly, we can consider the approach of establishing social networks through P2P overlays. Applications based on such overlays like Safebook and PeersON seems inspiring but its real time implementation is still to be evaluated [27]. LifeSocial provides a facility to add additional plugins for a better user interface and gives look and feel similar to the traditional OSNs. Additionally, the framework provides a secure and robust network for communication. In general, it is still important to consider whether P2P applications are used for file sharing, social networking, message passing, audio/video conferencing, multimedia streaming etc. The next generation of internet will demand to overcome the existing limitations of P2P networks. Some of the prevailing challenges are mechanism to find friends in a topology, robustness against churn, openness for third-party applications, storage of data, delay tolerant network, higher security against sybil and DoS attacks etc [21]. Resolving these issues when using P2P applications over a large network to provide a social networking framework can be a motivation

and open to further research.

4 Conclusion

Internet is an open platform which provides freedom to people to express their views and share them with the world. With the advancement of the web, more and more opportunities are created for social networking. But, with more and more advancing internet, privacy and freedom of people has also become a risk factor in oppressed countries. We have already discussed about the problems faced by the people under such authoritarian government. The idea of decentralization of data and anonymity is a counter attack on the actions of the oppressed regimes. People in countries with limited freedom of speech can be motivated to use current P2P applications and enjoy the free Internet. However, if tools are developed to fight against internet censorship, it is obvious that further tools will be designed against them. Its a never ending race, but still P2P technology has contributed remarkably in providing robust, private, secure and censorship-resistant applications. The future work in P2P technology can help to overcome majority of barriers on usage of internet, thereby become a key solution to existing problems.

References

[1] http://buddypress.org/about/.

[2] http://en.wikipedia.org/wiki/buddypress.

[3] http://en.wikipedia.org/wiki/foaf.

[4] http://gossple.fr/.

[5] http://ntrg.cs.tcd.ie/undergrad/4ba2.02/p2p/.

[6] https://we.riseup.net/crabgrass/about.

[7] https://www.psiphon.ca/.

[8] https://www.torproject.org/.

[9] http://uniteforclimate.org/.

[10] http://www.foaf-project.org/.

[11] http://www.freepastry.org/.

[12] http://www.helloworld-network.org/de/.

[13] wiki.debian.org/freedombox.

[14] www.diasporaproject.org/.

[15] M. Ackermann, K. Hymon, B. Ludwig, and K. Wilhelm. Helloworld: An Open Source, Distributed and Secure Social Network. In *W3C Workshop on the Future of Social Networking*, 2009.

[16] L. M. Aiello, M. Milanesio, G. Ruffo, and R. Schifanella. An Identity-based Approach to Secure P2P applications with Likir. *Peer-to-Peer Networking and Applications*, 4(4), 2011.

[17] T. Aydin and J.-P. Seifert. A P2P-Based Infrastructure for Censorship-Resistant Web Access. 2011.

[18] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. Characterizing User Behavior in Online Social Networks. In *Proceedings of ACM SIGCOMM conference on Internet measurement conference*, IMC '09, New York, NY, USA, 2009. ACM.

[19] M. Bertier, D. Frey, R. Guerraoui, A.-M. Kermarrec, and V. Leroy. The Gossple Anonymous Social Network. Springer Berlin Heidelberg.

[20] B. Beverly Yang and H. Garcia-Molina. Designing a Super-Peer Network. In *Data Engineering, 2003. Proceedings. International Conference on*, 2003.

[21] S. Buchegger and A. Datta. A Case for P2P Infrastructure for Social Networks-Opportunities and Challenges. In *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. International Conference on*. IEEE, 2009.

[22] D. Calingaert. Authoritarianism vs. the Internet. *Policy Review*, 160(63), 2010.

[23] H. Chen, X. Li, and J. Han. Maze: A Social Peer-to-Peer Networking. In *Proc. of IEEE International Conference on E-Commerce Technology for Dynamic E-Business*, 2004.

[24] H. Dabashi. *Iran, the Green Movement and the USA: The Fox and the Paradox*. Zed Books, 2010.

[25] A. Datta, S. Buchegger, L.-H. Vu, T. Strufe, and K. Rzadca. Decentralized Online Social Networks. In *Handbook of Social Network Technologies and Applications*. Springer, 2010.

[26] N. S. Good and A. Krekelberg. Usability and Privacy: A study of Kazaa P2P File-sharing. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2003.

[27] K. Graffi, C. Gross, D. Stingl, D. Hartung, A. Kovacevic, and R. Steinmetz. LifeSocial. KOM: A Secure and P2P-based Solution for Online Social Networks. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*. IEEE, 2011.

[28] R. J. Maly, J. Mischke, P. Kurtansky, and B. Stiller. Comparison of Centralized (Client-Server) and Decentralized (Peer-to-Peer) Networking. *Semester thesis, ETH Zurich, Zurich, Switzerland*, 2003.

[29] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Peer-to-Peer Systems*. Springer, 2002.

[30] A. Rowstron and P. Druschel. Storage Management and Caching in PAST, a Large-scale, Persistent Peer-to-Peer Storage Utility. In *ACM SIGOPS Operating Systems Review*, volume 35. ACM, 2001.

[31] T. J. Ryan. Infringement. com: RIAA v. Napster and the War Against Online Music Piracy. *Ariz. L. Rev.*, 44, 2002.

[32] Y. Sovran, A. Libonati, and J. Li. Pass It On: Social Networks Stymie Censors. In *Proc. of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2008.

[33] E. Stepanova. The role of information communication technologies in the arab spring. *PONARS Eurasia*, (15), 2011.

[34] C. von der Weth and A. Datta. COBS: Realizing Decentralized Infrastructure for Collaborative Browsing and Search. In *Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on*, 2011.

[35] P. Winter and J. R. Crandall. The Great Firewall of China. 2012.

[36] B. Woodcock and P. C. House. Overview of the Egyptian Internet Shutdown, 2011.

[37] C.-m. A. Yeung, I. Liccardi, K. Lu, O. Seneviratne, and T. Berners-Lee. Decentralization: The Future of Online Social Networking. In *W3C Workshop on the Future of Social Networking Position Papers*, volume 2, 2009.

Insecure monitoring: Aggregation attacks

Florian Schulte
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email: redeye@mail.upb.de

Abstract

In this paper i examine the security aspects of data distribution for monitoring and query purposes in the peer to peer application. For the currently used, tree based overlay implementation SkyEYE.KOM, a few weaknesses exist that can be exploited to forge information that is to be distributed across the network. The weaknesses mainly affect the aggregation of data, so a few methods are presented that can detect tampering of data. Beside this, an alternative method for data distribution is taken into consideration that works completely different than the tree approach by using a gossiping protocol. The techniques are compared regarding overhead and gained security. Also a few other vulnerabilities are treated that are only marginally associated with the actual aggregation attacks but have to be taken into account to ensure their effectiveness.

1 Introduction

1.1 Motivation

In peer to peer networks there is no central control instance by design. Normally this is not a problem because the nodes are managing themselves and for communication some routing technique is used. But there are a few cases where some knowledge about the whole network is needed. As already stated in [5] this information demand can be divided in two categories: On the one hand these are statistical information about the health of the network like average and maximum peer load or overall network performance.

On the other hand some capacity-based peer search is needed to decide how certain things regarding resource allocation. For instance, if a storage location for an object has to be determined, the node should choose a peer that meets some criteria like good availability and enough bandwidth to make the storage object accessible reliably and within a short time. To be able to do that, the current status of the nodes must be known to the network. While it is relatively inefficient to broadcast the data through the whole network, some approaches exist that use a more structured way to achieve this. One of them is SkyEYE.KOM which introduces an additional layer in form of a tree to the peer to peer system. As shown in [5] this tree approach is very efficient in respect to additional bandwidth consumption, but completely ignores considerations on security. Malicious or defective nodes can distort the data which will result in wrong decisions at a high probability. For instance it is possible to overload other nodes by untruthfully announcing that they have a big amount of resources. This also can be exploited by an attacker. Especially in the case of the p2p social network application, which claims that it can be used in oppressed countries, it is important that the network can't be shut down or impaired by individual users. In this paper some approaches are evaluated that prevent the attacks or at least keep the impact on the network at a low level.

1.1.1 Structure of this paper

In the next chapter the techniques used in the p2p framework are described. Based on this i show the weaknesses of these concepts in chapter 3, how attacks work and what their impacts are for the network. In the following chapter i show some approaches to prevent the vulnerabilities or at least make them much more difficult to exploit. Chapter 5 contains a comparision of the shown ideas and gives a suggestion which one to use. In the last chapter some final thoughts and few words on a discarded method are put down.

2 Explanation of used techniques

In the peer-to-peer application FreePastry[2] is used as the Distributed Hash Table[4] implementation. The application uses SkyEYE.KOM[5] as a monitoring instrument. It acts as an over-overlay on top of the DHT and forms a tree out of the nodes that are in the network. To do so, it uses the nodes unique IDs and recursively splits the ID space into multiple so called Domains. One node of the domain space becomes the coordinator. Normally this is done using a deterministic function, so each node in the domain can calculate the Coordinator ID locally. To avoid weak co-

ordinator nodes from being overloaded, a so called support peer can be defined which takes over the coordinator role.

In general there are two thresholds, T_{max} and T_{min} which define the maximum and minimum number of nodes that are managed by a coordinator. If the number of child nodes exceeds T_{max}, another level in the tree is formed. The T_{min} threshold ensures that there are never too few child nodes for a coordinator. This guarantees that there are not too many nodes a coordinator is responsible for. Also the tree is always at a reasonable height.

The idea is that the nodes send a message with the information about their status[1] towards the root node in predefined intervals. To prevent the root node from being overloaded by a large number of messages, the data of the child nodes is not transferred to it but to the respective coordinators one level up in the tree instead. The data reduction ensures that each coordinator does not get more than T_{max} messages per interval. In certain intervals the received data from the nodes of the next lower level are aggregated with some aggregation function like $SUM()$, $AVG()$, $MAX()$ or $MIN()$ and send to the parent. This procedure is iterated until the root node is reached. The root finally brings all records together and sends the final data set back to its subordinate nodes that distribute them down the tree on their own. This way the information for the statistical monitoring of the network this is available in the whole network and can be displayed easily by the network operators.

For the use case of capacity peer search it is necessary that the information about the resources is not aggregated. It is sent to the parent node and stored in a table as a key-value pair with the resource type as key and the origin node id linked to it. This information are merged for all descendants and again propagated to the higher levels.

If there is a node at an arbitrary location in the tree that wants to perform a certain task it sends a request to the upper levels of the tree. This request contains a set of requirements for one or more types of resources it needs and how many nodes of this type it needs. The request is handed over to the responsible parent node together with its own identifier. This parent looks up its table for nodes that can fulfill the requirements. If it finds a sufficient amount of nodes n, it answers with the list of matching nodes. Otherwise the request is iteratively delegated to the respective parent node that obviously has information about more nodes. If no coordinator can answer with a list of n nodes, the root node at least answers with as many nodes as possible.

3 Possible attacks or malfunctions and their impact

Having seen the motivation for attackers and the basic idea of network monitoring, now some strategies are presented that show how to exploit the lack of protection in the protocol.

While the used approach SkyEYE.KOM is very efficient regarding bandwidth overhead, it completely ignores security aspects. As the nodes are not operated by a central and trusted institution but by the users of the network, it is hard to control if data in the monitoring network is corrupted. This can either be the case because of some defective hard- or software or - more likely - because someone is manipulating the data on purpose. There are a few conceivable attacks on nodes at different key positions of the over-overlay tree. In particular i want to have a look at the coordinator nodes because they can affect the data of the most nodes.

3.1 Manipulation in a coordinator node

The easiest attack that can be performed by a coordinator node is to change the results of the aggregation function that is used for the values provided by the clients. For instance it is possible that the information about free capacity of the child nodes is being forged and then passed towards the root of the tree. This can lead to problems on the child nodes when they are chosen for more tasks than they can handle. Also for the overall monitoring SkyEYE.KOM yields wrong data. The overall impact very much depends on the position in the tree. In general the higher a malicious node is in the structure, the more values have to be considered as wrong. Beside that there are some special cases.

3.1.1 Root node compromising or downward propagation manipulation

With a limited probability the attacker has control over the root node of the overlay tree. Because all information converges at this point, this makes it easy to alter information for the whole network. The attacker can manipulate the data directly before it is sent down towards the leaves and there is no additional instance that can verify it. Fortunately it is very uncommon that an attacker becomes the root node by accident. It is more likely that an an intermediate node changes the data during the propagation phase even though fewer victims are affected.

3.2 Other weaknesses

Besides the aggregation manipulation in the coordinators it is possible that there are nodes in the network that provide incorrect values. Although this problem has nothing to do with the aggregation at first glance, it induces the same effects. Moreover, the coordinators are normal nodes from the point of view of the social network application. So they provide values themselves and can't be treated differently than the (pure) leaf nodes. This makes it even harder to detect aggregation attacks because the coordinators can always manipulate their aggregation results by changing their own values. Even if there is some verification where the

[1] Normally containing multiple ratios like cpu load, storage etc.

children nodes recalculate their result and compare them with the parent of the malicious coordinator can't know the correct value.

A thinkable reason to change the values is to make the node more attractive for data storage selection. Even if the malicious user can't know what is being stored, he can make some assumptions. For instance a rather big file and the way how other users access the file could be a hint that the file is a video that is streamed. Firstly this would make it easy for suppressive governments to find out who is distributing files. Also it makes it easier to remove or corrupt unwanted files if they are operating multiple of this nodes.

On the other hand a user can make his node more unattractive for network tasks when he doesn't want to share his resources. While this is not a direct attack, this parasitic behavior can affect the overall health of the network if done by a large number of people. This case is rather theoretical because the client allows defining how many resources are shared anyway.

Another case is that there is a node that is requesting his parent node for resources (e.g. the node needs storage space for a certain task). So it basically uses the peer capacity search provided by SkyEYE.KOM. The parent node could be acting corruptly and modify the request in a way that it is not fulfillable. The request is delegated up in the tree but even the root node can't provide host that have sufficient storage, so that the requester can't upload his file. This case can be relevant in oppressed countries where certain regimes want to prevent the users from distributing unwanted video material.

This effect becomes even more relevant when some algorithms are used that rearrange the DHT IDs in a way to map the geographical location to the p2p network to make it work more efficient.

4 Available Methods to avoid the attacks

To prevent the introduced attacks it is important that there is a mechanism that can detect that the data is compromised. Then in a second step, the nodes that are causing this, should be found. There are several approaches in the literature that show how to distribute data in tree structured or wireless peer-to-peer networks. It must be clear that there is no absolute security for the network simply because nobody knows how many of the nodes are malicious. If there is a corruption of multiple nodes that, on top of everything, can arrange with each other, detection the attackers would be very hard.

In the first part i take a look at some mechanisms that can be applied to the existing tree based approach without changing the actual overlay. In the second part i show a multi tree approach that reuses the concept of SkyEYE.KOM to detect manipulation. Thirdly a technique is presented that works apart from the tree approach using a gossiping protocol. The remaining two sections show ways how to deal with the side effects and shortcomings of the approaches.

4.1 Single tree approaches

The following techniques basically show how the data aggregation can be secured by verifying the values that are aggregated in a coordinator node. The basic problem also emerges in wireless sensor networks where different hardware elements send measured data of their surroundings. These devices are often operated in hostile environments and this is the reason why there exist some elaborations also in this disciplines of computer science. The difference is that the sensor networks have a fixed network structure at least in a certain degree while the over-overlay can be more or less completely rearranged. This additional flexibility offers some new opportunities to prevent certain things but also introduces different other attacks and shortcomings. For instance, in wireless networks real one-to-many messages are sent while broadcasting in the peer-to-peer network means sending single messages to all neighboring nodes, potentially with different content. On the other hand it is not so important that the communication between the nodes is encrypted, because no shared medium is used in the relevant layer of network protocol stack at least. Because of these and other differences regarding the application, the techniques presented only partially reflect the ideas shown in the papers. Only the very basic ideas were adapted for the scenario and were thought a step ahead. At the same time, i would like to point out that the adaptions are not practically tested.

4.1.1 Naive Approach: Additional Bypassing

The most basic method is to verify the data that is computed by a coordinator as shown in [7]. For now the cryptographic part of the protocol is omitted and shown separately. In this solution, the child nodes of a coordinator E at a certain tree level t_l are not only sending their data to the parent. They also use some kind of broadcasting for their tree level t_{l-1} to distribute the data between the siblings. Then either one arbitrarily chosen node or simply all of them perform the same operation as the coordinator E. Then the result is sent directly to the grandparent coordinator node F at t_{l+1} which also gets the value from the parent coordinator. Then it simply compares the values and knows if the data was manipulated during the aggregation phase.

Unfortunately this proceeding introduces a few new problems for the application. First of all, the fact is left out that the coordinator has an own value the siblings can't know, because only nodes on the same level are involved for the recalculation. This way the coordinator node could "repair" the verification value at least for some aggregation functions. This can be avoided if the coordinator does not

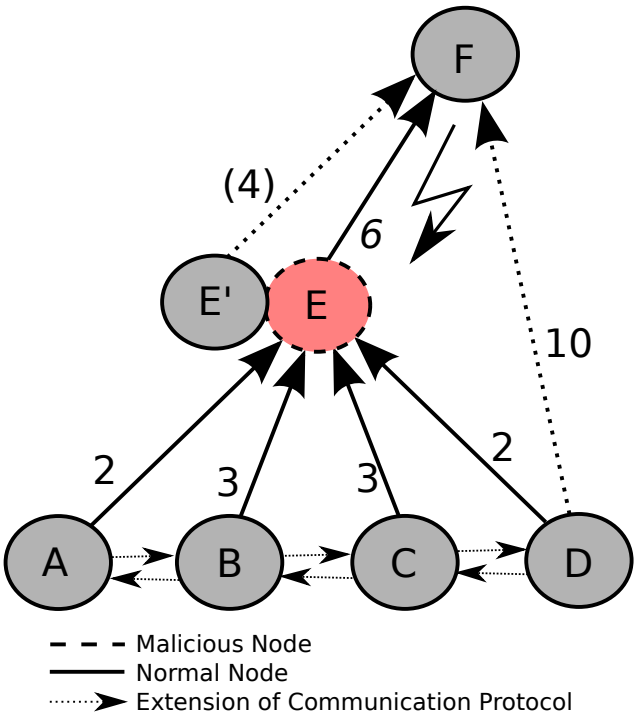


Figure 1. Bypass approach, F gets different values from D and E, correctly determines E as malicious

include its own value in the aggregation computation and sends them along with the result to the grandparent. This node considers both values as single children from level t_l, so the local part of E is considered as E'. It checks whether the result of the grandchildren's computations is consistent with the aggregated result from their parent and aggregates the values in the next step with the local value of E'. The problem of local value manipulation in this node still exists, but at least it is clear that the local value has to be verified which is easier. This is considered closer in section 4.4.

Beside this, some other special cases exist that have to be taken into account when this is applied for the peer-to-peer network as i will show in the next passage. Most of them can be fixed with some effort.

For the case that the data is changed at the root node, there is no further instance that can be used to verify the results. To avoid manipulation, the data that is being sent downwards the tree can be verified by the nodes in the first level. Because they are also exchanging data among each other, deviations can be detected. Of course this also becomes ineffective when even more levels are completely compromised.

The presented variant ensures that one single node that manipulates data can be recognized. If there are more malicious participants and they are direct ancestors, it's possible to circumvent the verification. It's very unlikely that an attacker can introduce two nodes that are in exactly this constellation. But for practical usage it must be concerned that a node potentially has the ip address of the parent and probably can get control over it if this serves other vulnerable services. To avoid this, the verification process should be adapted to be able to detect multiple malicious nodes. For this the values have to be transferred to the parent and the grandparent but also to its ancestor. This can be freely expanded to more levels, but introduces the problem of multiplied traffic amount for the higher nodes.

Another Problem that exists is that a direct descendant of a coordinator that itself is innocent, can spread different values to the parent node. This way another kind of aggregation or value changing attack is introduced. This is demonstrated in figure 2. In this example, node E has sent its value 2 to the parent node and now aggregates the values of its siblings A, B and C to send them to the grandparent node F for verification. The difference is being noticed by F and an aggregation attack is assumed. For this case it is not helpful to penalize D by excluding it from the network. A few notes how to deal with potentially malicious nodes is demonstrated in the chapter about attacker handling (section 4.5).

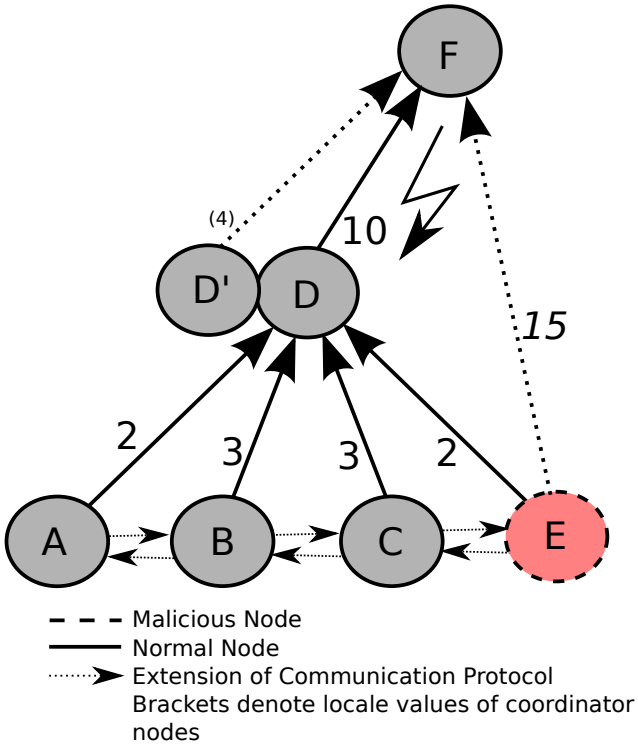


Figure 2. Example of a child node yielding wrong values for SUM() to grandparent

For example, if one of the leaf nodes provides different values to his parent and to the grandparent for the verification, the aggregator would be considered as malicious while it didn't do anything wrong.

Now i take a look at the resource overhead for this method. For the aggregatable data in original method, each node sends the information once per interval. So there are T_{max} messages per node and $logN \cdot T_{max}$ for the whole network at most. As the additional bypassing is introduced, there are another T_{max} messages at least for all child coordinators below, summed up $O(T_{max}^2)$ instead of $O(T_{max})$. In addition the communication between the siblings must be taken into account. For the simple case where only one node calculates the value, this means one additional messages per node. To reduce the overhead, the verification can be made optional so that it's only performed with a certain probability.

4.1.2 Cryptographic Signatures

The following method can be understood as an extension of the previous with the difference that it uses some basic cryptography instead of bypassing. It is very similar to the method described in [7].

Instead of sending the plain information to the parent node, in this approach all direct descendants $N_{1..k}$ of a potentially evil coordinator C, sign the data. For this, their private key is used that each node has anyway in the peer to peer application. When the messages arrive at C, the data is aggregated and sent to the next parent as in the original implementation. Additionally the k signed messages are sent upwards for verification. This way it is not necessary for them to send the data multiple times and this also eliminates the problem that a node N_i provides different values to the parent and for the verification which will result in a parent denunciation.

Along with the actual data, the sender id, a nonce[2], a TTL value and the current time or time interval number have to be included. The TTL value t is specified for the whole network and regulates how often the original messages are sent one level higher in a non-aggregated form. The TTL value is not part of the signed data because it is decreased at each parent hop until it becomes 0. After this, the message is not forwarded anymore. This means, all coordinators that are not more than t levels away from the origin node, can verify if aggregation was done correctly. As the nonce is used and verifier stores all nonces for a certain period of time, it can be detected if a coordinator is sending a message twice. The time value is used to expire old nonces. This way the amount of nonces that have to be stored can be kept constant.

This approach allows to specify much easier how many consecutive malicious nodes can be detected by specifying the TTL.

The drawback is that the verifying nodes must be aware of the number of the grandchildren and deeper descendants. Otherwise malicious nodes can simply drop the messages to completely hide the node from the upper coordinators. If necessary this can be done with some keep-alive packages sent directly to the grandparents that makes this approach less convincing as it introduces the same message paths as the non-cryptographic version.

4.1.3 Multiple Parent Method

While in the naive bypass approach the children nodes are responsible for redundant data collection, this task can also be delegated to independent additional parent nodes that have the same status as the original one. For this, a predefined redundancy parameter is specified that describes the number of parents p for each level of the tree that are responsible for the same domain. The additional parent nodes have to be selected by using the p nearest neighbors of the original parent node for a domain. This way the originial tree structure is degenerated, but this does not introduce any problems as the communication for the operations is always directed and no cycles are constructed.

All information from the lower level is sent to all p parents. All of them calculate the result for the aggregation and send them to the parents at the next level in the tree. Their local values are still sent individually to keep the aggregation results comparable. An example that shows the principle of the communication channels is shown in figure 3. A, B and C are leaf nodes here that only send values while E, F and G aggregate this values and send them to H. In the example E yields an incorrect result to H. This node can now figure out which senders belong together because they always belong to the same domain range if they are responsible for it. In case one or more of the values differ, some tampering has taken place. With this method, in contrast to the previous, it is possible to continue with the correct values computation with a high probability. For the aggregation, one of the values has to be chosen. The best strategy for this is to use the result that most of the nodes yielded. In the example F and G are working correctly and therefore H notices that E has sent a forged value. Even if multiple nodes provide incorrect values, it is very unlikely that they manipulate the data in the same way as long as the majority of descendants deliver correct values. Of course it is necessary to predefine an appropriate value for the number of parents per node. A higher constant provides better security but also results in an increased redundancy factor.

The number of messages that have to be send by each

[2]Number used once, a value that is different for each message to make the message and the signature uniqe. Used to prevent replaying of messages.

node grows by the factor p, but at least it's non-varying for all the tree levels. This prevents an overloading of the root node.

At first glance for this approach the same attack as for the naive approach are possible. One of the value yielding nodes can send different values to the parents. While in the previous method this was possible for values that were aggregated and for the local data, this applies to the local values only in this method. For this value changing attack this cannot really be prevented but at least it is possible to limit the effect as shown in section 4.4.2.

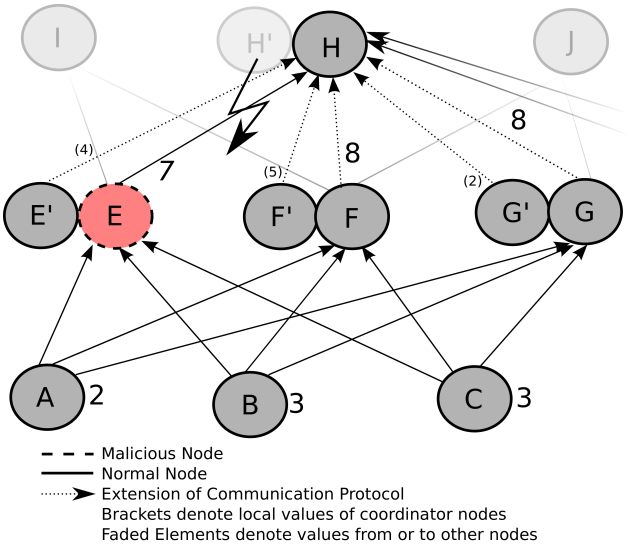


Figure 3. Example of Multi Parent Method

4.2 Multiple tree approach

For the following solution i do not extend the concept of SkyEYE.KOM with some kind of cryptographic or bypass verification, but simply reuse the concept and apply multiple trees to the peer-to-peer network.

As shown in section 2 the assignment of the root node is based on the key space of the DHT. While the standard implementation simply takes the middle element from the range, it is easy to build a tree that uses another node as its root. As the free pastry DHT implementation uses a cyclic allocation of ids anyway, there is no problem with the domains here. Some kind of shifting parameter can be passed that indicate the distance to the standard values for the responsibility function. This way it's possible to choose additional nodes from the key space and create a tree with them as particular root. The original tree is denoted as T_0 and the new trees as $T_{1..k}, k \in \mathbb{N}$.

Now a malicious coordinator node E is assumed that is at a certain position in tree T_0 which is forging aggregation values. As there is no verification, this cannot be detected and the root of T_0 will propagate the falsified statistic down the tree. If another tree is added now, chances are very low that the malicious coordinator will have the same child nodes as in T_0. Most likely it even won't be at the same level of the tree, so the impact of its changes is completely different. This means that in case of a manipulation, the root nodes of the trees will get different values. The root nodes send the calculation down their tree in combination with their tree ID. If an arbitrary node receives different values for the trees, a potential aggregation attack is revealed.

This approach seems to be very elegant as the additional overhead compared to the original way increases linearly with the number of trees, but there are some drawbacks.

Because the nodes most likely have different positions with a different level in each tree, their values are not aggregated at the same time. The reason is that it takes some time until the data is aggregated and transferred to the next hop in the tree. The following case must be considered: A node that has submitted its value to its coordinator for tree T_0 and is shut down before it was able to send the data to the belonging coordinator for tree T_1. Here the values for T_0 and T_1 differ. This effect can be counteracted by an observation of the values for a longer time, but the bigger the network becomes, the more likely it is that one of the nodes joins or leaves.

Also it is not so easy to find out which node introduced the forged data. The deviation in the values can be taken as an occassion to do some deeper inspection. As there are multiple trees, every root node in a tree is one of the descendant nodes in another tree. Therefore there is always a tree that has the attacking node in one of its subtrees. To be sure to get a tree where the malicious node is not root, a tree or respectively its root is randomly choosen. The idea is to identify the subtree that has the malicious node in it. To do so, another tree is formed that uses the same top level domain arrangement but with another root node. For a network with circular IDs, this is the "opposite side" node of the original root. They both have the same participants in their subtrees but with a different arrangement. After the next round of aggregation the nodes will get different values for one subtree or domain if the node is still manipulating. This can be done recursively for the conspicuous subtree until a single node is identified which is the evil attendee.

Because this detection is very complicated and inefficient, it is not considered any further.

4.3 Alternative that does not use trees: PushSum

In this part i want to present an approach that does not use tree based concept for data distribution. It's called PushSum and was published by Kempe, Dobra and Gehrke[6]. It is not as efficient as the original SkyEYE.KOM implementation and therefore was not worth to be considered before. But if we take the measurable overhead and the increased

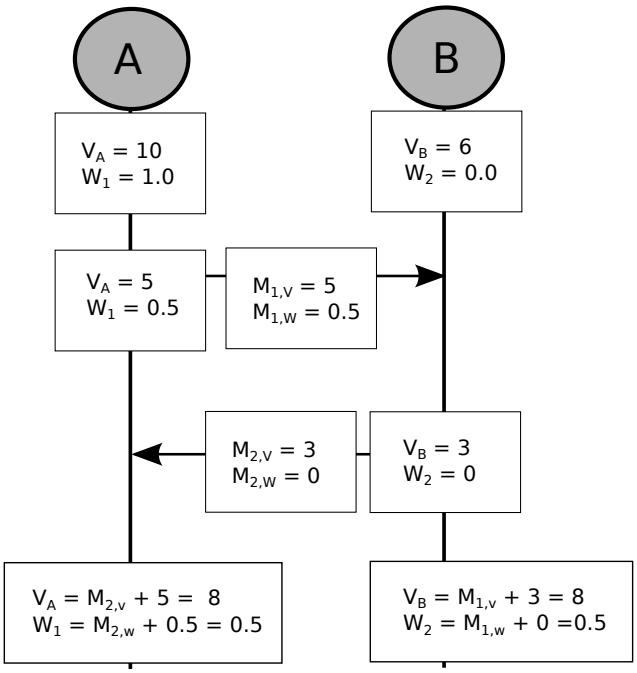


Figure 4. PushSum approach Example for $SUM()$

complexity into account that is needed for the safeguarding of the aggregation now, it may become a good alternative.

For the aggregation of $SUM()$ and $AVG()$ the algorithms are very similar. It's described as follows in [6] and is presented in a more vivid way at [1]: The basic idea is that each node i stores a sum $v_{i,t}$ for each round t. This sum is initialized with the nodes' own value. The weight $w_{i,t}$ is used for counting. The values are distributed across the network so that the weight converges against the number of nodes n involved for $AVG()$ and against $\frac{1}{n}$ for SUM() aggregation.

In each round, each node receives at least one message that were sent in the previous round. The messages contain a pair of a value s and weight w. Both of this fields are summed up for all messages received in that round and stored locally.

Then one of the neighbors is picked in a random manner and the locally stored values multiplied with a factor of 0.5 are sent to the choosen node and to the node itself to be used in the next round.

For the first round the local value of the node is sent to itself for initialization.

To calculate the estimated outcome using the local values of node i in round t, the $v_{i,t}$ has to be divided by the weight $(\frac{v_{i,t}}{w_{i,t}})$.

Figure 4 shows an example for the algorithm with two nodes for the first round of the $SUM()$ aggregations. A sends its values halved to B. This node does exactly the same. In the end, both nodes have 8 stored as the value and 0.5 for weight, so the fraction yields 16 as the overall sum.

Because the peer capacity search functionality must be available even when the gossiping approach is used instead of SkyEYE.KOM, a functionality has to be provided. A search for a predefined amount of nodes with certain criteria can be performed using a broadcast request. This also contains the original requester id and perhaps a time-to-live value to limit the spreading. Each matching node will directly send an answer to the requester.

All in all this protocol yields good results and runtimes as indicated in [6]. On the other hand it is a probablistic algorithm and uses a completely different concept than the tree approaches.

4.4 Proposal how to avoid the other attacks

In this part i show some ways how to prevent the attacks that were nominated in section 3.2 or have arisen for the presented methods.

4.4.1 Propagation data manipulation

Until now i only considered the data when it is sent towards the root node. For the SkyEYE.KOM Protocol the other way around is just as important. Fortunately the propagated data tampering can be avoided relatively easy, as all the nodes in the peer to peer network have a public and private key pair, which can be used to thwart this. The root node creates a hash over the information that is sent to all tree members including a nonce value like it is used in section 4.1.2. This hash is signed using the nodes private key just like signing an email. Both, the signature and the information are handed over to the descendants. In case of modification, the signature verification would fail and the message is not sent any further. This ensures that violation can always be associated with the current parent node. Also a replay attack is efficiently prevented because of the nonce. For this the root node must be trusted, otherwise it can forge the aggregation as well. It might make sense use a root node that is under control of the operators of the network as an exeception to the principle of equality.

4.4.2 Value Changing

As shown there are some value changing attacks that are not directly involved with the aggregation but can't be ignored either. They can have nearly the same impact on the network and as this paper tries to show some practical solutions, it shouldn't be completely ignored. So here i present a

few suggestions how some of the measurements can be verified with a proof-of-work approach. For example, if a node pretends that it has a lot of storage, probability is high that it will be chosen to store data. As shown earlier, it is possible for an attacker to take advantage of this. If a node provides some unusual high values it can be chosen to verify that the indicated storage capacity is correct. To do so a cryptoraphic task can be handed out to the node before its values are accepted by a node that is responsible for the node, i.e. its parent in the tree based protocols. One way to do so is to make the node to generate and store an appropriate amount of strings plus some kind of one-way-hash values for them, so that the full capacity is needed. After this precomputation is finished, the verifier generates some random strings from that range itself, computes the hash and sends these hashes to the subject. If it can answer with the correlating strings within a short amount of time, it must have saved the entries and therefore must be able to store the pretended amount of data. To verify cpu power or bandwidth, similar tasks can be used.

These tests have a few drawbacks. They are very time-consuming for the local node and can only limit the values either to a minimum or maximum so they should be used very rarely. At least they are sufficient to find the value changing outliers with extreme discrepancies.

4.4.3 Capacity peer search request manipulation

For this attack the shown aggregation protections can be adapted. The request can either be signed with some cryptographic signature for the Cryptographic Method or simply be sent to all parents for multi parent approach. In case of manipulation this can be detected by verifying the signature for the first case. In case the message was completely dropped by a bad coordinator this can be realized as no answer is coming from the root node.

In the multi parent approach some kind of request ID is needed to reassign multiple requests in the nodes that receive multiple nodes.

4.5 How to deal with malicious nodes?

After identifying a malicious node some kind of intervention is needed. It is possible to entirely exclude the node from the network, but there are some cases to be considered in case the real manipulation has not taken place in the aggregator. For example at least when using the naive approach, the possibility exists that a node can denunciate its parent node as malicious as explained in figure 2. A better way would be to reorganize the tree and add some kind of reputation system. a potentially malicious node could be moved to a lower position in the tree and it could get some kind of reliability rating. If the node was the faulty one and it still tries to manipulate the network, this will at least have a lower impact on the network because of the lower position in the tree. If the identified node is not malicious one, it has new nodes around and the reputation will rehabilitate again over time.

5 Suggestion how to solve the insecure monitoring problem for the application

In this section a summarizing comparison for the shown approaches is constituted and a final suggestion is given.

The naive bypassing approach gave a good starting point and makes the original implementation more secure but besides this has some drawbacks as shown in table 1.

Also the multi-tree variant is not a good choice because when networks are getting larger the probability that at least one participant leaves or joins the network during one aggregation phase becomes very high. If the leaving node is counted in the one tree but left before it has been counted for another tree, values differ. In this case a false positive alarm would be raised.

The other methods look more promising. For usage with SkyEYE.KOM the multi-parent-variant performs a bit better because it keeps the additional message overhead constant and doesn't overload for the upper level nodes with too many messages per round. Also it does not only detect the malicious node but also keeps the monitoring intact with high probability.

For usage without the tree overlay, PushSUM is a good choice for aggregation. Because all nodes are equal here, the impact of attacks is relatively low. On the other Hand its even harder to detect manipulations for it. Especially local value manipulation can hardly be detected because there is no verification instance that can be used.

6 Conclusion

In this paper i have presented a few thinkable attacks on the aggregation of the SkyEYE.KOM over-overlay. Their impact on the peer-to-peer network was shown and the results for the application were demonstrated. It turned out that the good efficiency of the original tree overlay is reduced when trying to provide additional security for the protocol. It is possible to increase the security with some effort and at least the runtime is not significantly affected by the most of the methods.

While several possibilities have been found that can make the manipulation of the aggregator functions a lot harder, it turned out that it is not easy to overcome all thinkable attacks, especially when local values of single nodes are changed in a way which cause certain effects on the network. To restrict this attack vector at least, a few suggestions are presented that use some proof-of-work concept to validate the values.

In the end, there must be a reasonable relation between additional overhead and gain of security. The methods offer possibilities to adjust this, basically by defining the maximum number of co-working attackers that can be detected.

	Naive Bypassing Approach	Cryptographic Aggregation	Multi Parent Method	Multiple tree	PushSum
Benefits	–	• security gain controllable by TTL value	• security gain controllable by defining amount of redundancy • message overhead constant for all levels in the tree	• easy to detect aggregation manipulation	• simple and robust protocol
Drawbacks	• additional aggregation weakness introduced when data for verification is collected by a single node • potentially increased runtime for data collection	• message overhead in upper levels	• changes in routing protocol necessary to find additional parents	• hard to find actual attackers • bad for bigger networks	• additional method for capacity based peer search needed • hard to predict message flow because of random character

Table 1. Confrontation of the different approaches

In combination with the protection against propagation manipulation as shown in section 4.4.1 this should provide good results.

Although this is not absolutely secure, it must be clear that it doesn't make sense to secure a network to all extend that is permeated by malicious nodes. This is especially true when other ways to attack the network exist that are easier to exploit or cannot even be straightened out.

Another concept that is not presented here but was looking very promising is the use of homomorphic encryption for tree aggregation protection as shown in [3]. It allows performing mathematical operations on encrypted data. This way it would have been possible to let the lower nodes encrypt their data and some kind of verification key. The coordinators aggregate the encrypted data without even knowing the actual values. The outcome can be decrypted by some node in the higher levels. In case of manipulation the data would be either garbled or does not match to the verification key. The problem with this approach is that total homomorphic encryption is rather new. It either yields correct values only for a couple of computations as there is some error involved that gets too big or requires a lot of precomputed data with current implementations. Maybe this is worth a reconsiderationin the future when it is matured.

References

[1] Intelligent Network Part III Tree Aggregation and Symmetric Push Sum Protocol (Gossip Protocol). `http://www.aidriven.com/index.php/tag/symmetric-push-sum-protocol/`, 2012. [Online; accessed 28-April-2013].

[2] Pastry - A scalable, decentralized, self-organizing and fault-tolerant substrate for peer-to-peer applications. `http://www.freepastry.org/`, 2013. [Online; accessed 14-June-2013].

[3] C. Castelluccia, A. C.-F. Chan, E. Mykletun, and G. Tsudik. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. ACM, 2009.

[4] F. Dabek, B. Y. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a Common API for Structured Peer-to-Peer Overlays. In *Proceedings of the International Conference on IPTPS*. Springer, 2003.

[5] K. Graffi, A. Kovacevic, S. Xiao, and R. Steinmetz. SkyEye.KOM: An Information Management Over-Overlay for Getting the Oracle View on Structured P2P Systems. In *Proceedings of the International Conference on ICPADS*. IEEE, 2008.

[6] D. Kempe, A. Dobra, and J. Gehrke. Gossip-Based Computation of Aggregate Information. In *Proceedings of the International Conference on FOCS*. IEEE, 2003.

[7] H. Li, K. Li, W. Qu, and I. Stojmenovic. Secure and Energy-Efficient Data Aggregation with Malicious Aggregator Identification in Wireless Sensor Networks. In *Proceedings of the International Conference on ICA3PP (1)*, 2011.

Overlay heterogeneity and capacity-awareness: Support for mobile nodes and smartphones

Navendu Barua
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email: nbarua@mail.upb.de

Abstract

Peer-to-peer paradigm is on the bloom due to their fault-tolerance, scalability and self-organizing nature, Napster and Gnutella are the best example. And as the current trend is going more towards wireless technology more specifically for mobile/smartphones there is a need of an approach to integrate Peer-to-Peer (P2P) paradigm into mobile/smartphones. Currently this integration exists, but only for homogeneous system i.e. all the mobile nodes/smart phones are considered to be alike. But for P2P to work more efficiently it should be able to distinguish between smartphones on the basis of resources and provide services accordingly.

This paper focuses on various approaches like Chordella, Superpeers, Distributed Hash Tables (DHTs), Chord, Content-Addressable Network (CAN), Pastry etc. for Peer-to-Peer concept to achieve this heterogeneity.

1 Introduction

With Napster and Gnutella systems, peer-to-peer networking has become quite famous. Although in both Napster and Gnutella peer-to-peer networking concept was used only to share files (audio-mp3 format), it can also be used for multimedia communication networks based on Peer-to-Peer concepts or resource sharing.

What is Peer-to-Peer network [1]?: A distributed network architecture may be called a Peer-to-Peer network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers ...). These shared resources are necessary to provide the service and content offered by the network (e.g. file sharing or shared workspaces for collaboration): They are accessible by other peers directly, without passing intermediary entities. The participants of such a network are thus resource providers as well as resource requestors (Servent-concept).

Currently all the nodes in peer-to-peer network are considered being equal i.e. same roles, duties, also performs the same task. But in a heterogeneous system the nodes can have different resources, the user might want to access the network based upon these resources (bandwidth, storages etc). For these types of requests there is a dedicated virtual network on top of the physical telecommunication network and it is called as Overlay Networks.

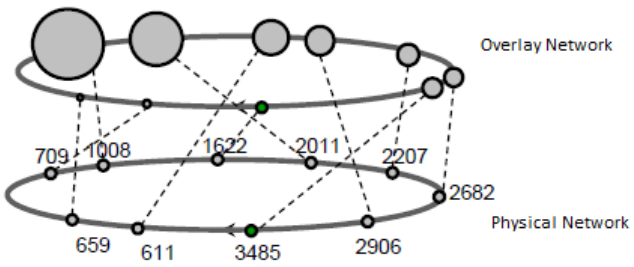


Figure 1: Overlay Network on top of a Physical Telecommunication Network[2]

This Overlay Network is of two kinds: Structured P2P and Unstructured P2P. In a structured P2P architecture, a DHT algorithm is used to locate the content in the network-each node/device is given a unique ID and each data item is assigned a file ID [3]. Each node/device has a pre-defined range of the file ID space. CAN, Chord, Pastry are some of the architectures that uses this kind of architecture. These lookup algorithms in structured P2P systems assume all the nodes uniform i.e. with same resources like network bandwidth, storage and CPU processing load. And in an Unstructured P2P Network the links between peers are established arbitrarily. Gnutella, Kaaza, Freenet, BitTorrent, etc. are some of the examples of this type of network.

The outline of the paper is as follows: Chapter 2 will focus on challenges and issues of deploying P2P for mobile phones. Chapter 3 will be covering the fundamental concepts, which will be useful to understand the different approaches discussed in further chapter. Chapter 4 discusses

different approaches for Structured P2P network for mobile nodes/ smart phones with different capacity. And chapter 5 focuses on the issues of using desktop application directly in mobile nodes/ Smartphones and also discusses a cross compiling language for the same.

2 Challenges and Issues

There have been various challenges and issues with integration of P2P paradigm in mobile network. The main task in Peer-to-Peer network is sharing of resources of one node with other nodes in the corresponding Overlay Network and same is there for mobile phones as well. But doing so might face some challenges in mobile phones like: the mobile nodes need to be robust in essence of failure of a peer and also there is a high churn rates with mobile nodes i.e. users leave and joins the network at a very high rate. As discussed in [4], in P2P solutions often flood the participating peers with request, which results in inefficient overhead. But there has been development in overcoming this issue using concepts like Chord which can resolve the query whether the requested resource is available or not.

Scalability is another challenge for P2P in mobile networks. Mobile networks are usually limited in their resources. They have lower data rates, low processing power and storage capacity. Also there is a high churn rate issue i.e. the rate of joining and leaving of peers from the network which could result in peer failures (e.g. moving out of coverage, discharged batteries etc.). If this P2P paradigm has to be implemented in mobile networks, the high heterogeneous environment in mobile networks than the usual desktop PCs in fixed Internet P2P networks have to be considered.

Energy Consumption is also a challenging issue in mobile devices. The developer's main concern is to develop the mobile application in energy optimized way. There have been various approaches focusing on this challenge like in [5] the researchers proposed an energy optimized, load balancing protocol for Kademlia based DHT which allows mobile devices to reduce energy consumption by selectively dropping incoming messages over WLAN. Some of the challenges for P2P concepts for mobile network from [6]:

Bandwidth Constraints: The P2P overlays designed for the wired Internet rely on the IP routing infrastructure which is resource rich especially in terms of bandwidth availability. Therefore there could be scalability problems because of high maintenance traffic when legacy P2P services are used in multi-hop environment. There have been various approaches to overcome this challenge by integrating or applying cross-layering techniques between the P2P and the mobile ad-hoc network (MANET) routing layer.

P2P Overlay Maintenance: One of the main tasks of a DHT system is to keep the routing table of each node up to date. For efficient routing, the routing information has to be current and consistent. Invalid entries cause unnecessary overhead. To avoid these inconsistencies, DHT protocols employ maintenance mechanisms to keep the routing tables up to date. There have been cross-layer and integrated approaches like CrossROAD which exploits the network routing messages or cache information in order to maintain the P2P overlay.

Exploiting Heterogeneity: Taking heterogeneity into account is also one of the important challenges for P2P concept as it has to decide which nodes should be there in overlay as not all the nodes in a network can become overlay members. Initially all the nodes in an overlay are placed manually but later the nodes can decide to join and make services available dynamically or automatically. There can be other constraints which can drive the decision to participate in the overlay. For example, nodes with limited power may not wish to act as overlay routers for other nodes.

3 Preliminaries

A hash table is a data structure that maps keys on to values efficiently. It is the essential building block in the implementation of software systems. In [7] a **Distributed Hash Tables (DHTs)** has been described as distributed data structures that use hash functions to index items that should be located on the overlays. Nodes and the data are mapped on the same address space i.e. they have the flat identifier, 1-dimensional IDs with common address space. The aim of DHT based systems is to provide efficient way of routing the messages to their logical destinations. In most of the approaches, the cost of this routing mechanism grows logarithmically with respect to the size of the overlay network. DHTs have been the basic block in many approaches like Chord, CAN and Pastry.

Chord [8], is a distributed lookup protocol that addresses the problem of efficiently locating the node that stores a particular data item. In Chord, a key is given to the data which is mapped onto a node and this key/data item pair is stored at the node to which the key maps. The main features of Chord are its simplicity, provable correctness, and provable performance even during high churn rate. Its performance can be degraded if the information of node is partially correct. The various approaches that follow Chord include Chordella, MeshChord, Hybrid Chord Protocol (HCP).

Pastry [9] is a DHT based systems, presents application with a homogeneous structured application-level overlay network. Pastry is completely decentralized, fault-resilient, scalable, and reliable. A message in Pastry is routed to the live node with a nodeId numerically closest to a key. Each Pastry node keeps track of its immediate neighbors in the nodeId space thus this helps in tracking arrival of new nodes, any node failure and recoveries. Pastry can be used as a building block in the construction of a variety of peer-to-peer Internet applications like global file sharing, file stora-

ge, group communication and naming systems. Pastry routes to any node in the overlay network in O(log N) steps in the absence of recent node failures, and it maintains routing tables with only O(log N) entries. Moreover, Pastry takes into account locality when routing messages. Results with as many as 100,000 nodes in an emulated network confirm that Pastry is efficient and scales well, that it is self-organizing and can gracefully adapt to node failures, and that it has good locality properties.

Content Addressable Network (CAN) [10] is a distributed infrastructure that provides hash table like functionality on Internet. It is scalable, fault tolerant and completely self-organizing. It has a structured P2P overlay like Chord but it has d dimensional torus. The entire coordinate space is partitioned and shared among the CAN nodes. A distinct rectangular zone is assigned to each node. CAN nodes stores all the files that hash into its zone. Figure 2 shows a 2-dimensional coordinate space partitioned between 5 CAN nodes.

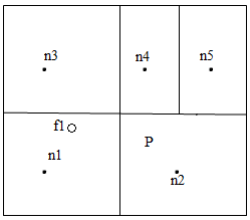


Figure 2: 2-D Coordinate space of CAN

There are pairs of keys and values that get stored using this coordinate space. Key is deterministically mapped onto a point P in the coordinate space with the help of a uniform hash function. This pair of key and value is then stored at the node that owns the zone within which the point P lies. Every node knows the IP addresses of its neighbors and the coordinates of the neighboring zones. And if some file is not found within CAN node zone than a node can interact with its neighboring nodes to look for the file in their particular zone which then forwards the request further if the file is not found.

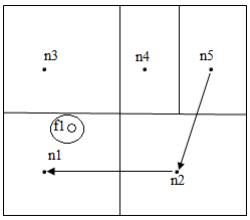


Figure 3: Routing

For example, n5 is looking for a file f1, which is present in node n1. Node n5 has 3 neighbors: n2, n4 and n3 (remember the space is d-dimensional but we can show it 2-dimensionally only). Node n5 can route the message request to any of its neighbor depending upon which protocol is getting followed. In this example the request is forwarded to n2. Node n2 looks into its zone for the file f1, which is not present and hence the request is further sent to either n4 or n1. In our case n1 receives the request and file f1 is accessed by node n5. The scenario is represented in figure 3. If a new node comes into the system the space is adjusted as follows: A new CAN node first discovers an IP address of one the node present in the system. CAN uses a bootstrap method described in YOID [11] for this purpose. The node randomly chooses a point P in the space and route to the node responsible for the zone of, shown in figure 4.

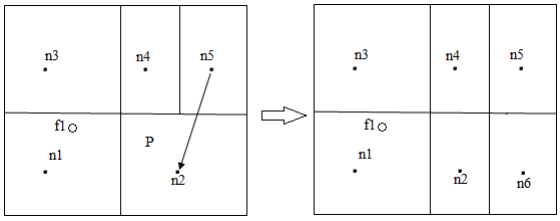


Figure 4: Joining of a new node.

The zone in which P lies gets splits in half and assigns one half to the new node. For a 2-d space, the zone first splits along x-axis and then the y-axis. Content (key, value); corresponding to the new space is also transferred to the new node along with the information about the neighbors from the previous occupant. In order to update their neighbor sets, all the nodes in the system then sends an update message, followed by periodic refreshes to all its neighbors.

In case a node leaves/fails, the zone of the node can be merged to a neighbors zone producing a valid single zone. If the producing zone is not valid, then the zone is given to the neighbor with smallest zone.

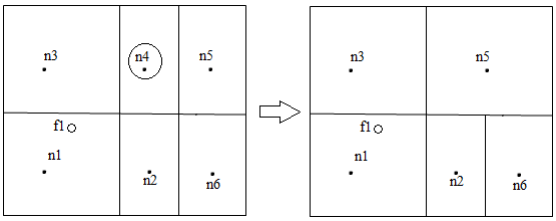


Figure 5: Node n4 leaves the system.

Even though lookup in CAN will find the target (if it exists), the lookup performance is still not good. Hashline is one of the example where CAN is used.

4 Related work on P2P paradigm and Mobile networks

Hashline [6] , a DHT-based file sharing system for wireless ad-hoc networks. It integrates the P2P query functionality with network routing. Hashline is based on the adaptation of the CAN [10] P2P routing protocol with a difference of use of a 1-D space called hashline, in which mapping of

node IDs and keys is established. Location queries can be answered through Hashline and also it can be used to discover and maintain routing information that is used to transfer files from a source peer to another peer. And hence the proposed P2P file sharing system can run on an ad-hoc collection of wireless nodes without requiring a separate mobile ad-hoc network **(MANET)** routing protocol at the network layer. The hashline is divided hierarchically into segments so that each node is responsible for one segment. The values of the keys present in the segment are stored in the corresponding node responsible for that segment. The authors of [12] describe the routing technique for the approach. The relationship among segments is like a tree with parents and children so that the hashline segment of a parent can span the hashline segments of all its children. When a node tries to locate a file with key k, if the key k is in the hashline segment of one of the nodes children the query is forwarded to that child else the query is forwarded to the parent. And at the end, the node responsible for the hashline segment with the key k receives the query and then node knows the location of the file and the route to that location as well. However, the approach is not suitable for the system where there are frequent disconnections. The idea of Hashline was proposed with an assumption that the environment used for sharing the file is wireless ad-hoc network (WANET). This cannot be used for a system with high mobility rate. For mobile phones this can never be assumed, as they have a very high mobility rate. Mobile phones come and go from a network very frequently depending upon the area they are physically present. So, if P2P has to be adapted for mobile phones, using Hashline might not be the best approach to follow.

In [13], the researcher has proposed a new system- **Chordella**. Chordella is a 2 tier P2P overlay network system. The architecture of Chordella consists of two instances: a fully featured C++ based peer application using the same codebase as the simulations used for Chordella (**Super-peer**) and a lightweight, mobile peer application running on mobile phones (**Leaf-node**). Low level nodes (leafnodes) - These will comprise of resource-constrained wireless devices such as mobile phones. They do not need to provide high data rates nor do they require high computational power or a certain uptime for their participation in the P2P overlay network. Upper level nodes (superpeer) - They have substantial resources and have a more reliable network connection. They create a ring-shaped structured DHT overlay based on Chord. Leafnodes do not participate in overlay; rather they are point-to-point connected to a superpeer. And because of this they do not share the burden of possibly high maintenance traffic and the overlay network does not have to deal with their performance bottlenecks and low reliability.

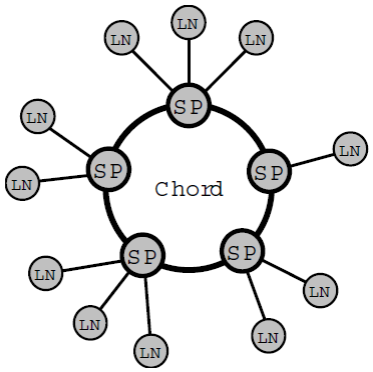


Figure 6: Hierarchical structure of the Chordella system[13]

The fully featured peer application can work as Superpeer and as a Leaf-node. As a superpeer it takes part in the DHT based overlay network and act as a proxy for the leafnodes it connects to. And in addition to the basic DHT features there is **Load Balancing Algorithm**, which ensures the same load levels of the participating super-peers; **Optimal Operation Point Selection Algorithm** dynamically adjusts the number of superpeers to a cost-optimal value. Suppose some superpeer fails due to the overload. Then using this algorithm the failed superpeer will make one or more of its leafnodes as superpeer thereby decreasing the load and making the system stable again. And there is also an integrated **Caching Algorithm** for improving the system performance and reliability by adaptively caching references to popular contents. One of the major challenges in structured P2P system is providing an efficient lookup and routing algorithm. In all these algorithms there is always an assumption of all the nodes having the same resources (network bandwidth, CPU storage and memory), which is not true in reality, and if the participating peers are mobile phones then the resources can vary more than the fixed peers. Researches like [14] proves heterogeneity as extreme in the deployed P2P systems (for example with up to 3 order of magnitude difference in bandwidth). The limited resources of some peers can result in a bottleneck which in turn could lead to the inefficiency of the existing location and routing algorithms.

In [15], a **Super-Peer** based lookup in structured p2p systems, authors propose an approach to improve the efficiency of these lookup algorithms by exploiting the heterogeneity (network bandwidth, storage capacity and processing power) among participating mobile nodes. The algorithm assigns greater responsibility to the superpeers having high network bandwidth, large storage capacity and significant processing power. The superpeer lookup algorithm and existing lookup algorithm can exists simultaneously using a hybrid approach. In this case first the superpeer lookup algorithm is used and if the lookup fails to route the query message then the existing one is used for resolving the que-

ry. In this approach, another Super-Peer Overlay gets added on top of the original Overlay. It consists of only Super-peers. Now there are three roles for a super-peer: Being peer on the original overlay, on the secondary overlay and being as a centralized server to a set of week peers present on the original overlay (also called as Clients). These clients sends queries to their respective super-peer and receive results from it. [15] named these pair of super-peer and client as a cluster as in Figure 7 where black nodes represent super-peers while white nodes represent clients.

A super-peer has 2 node IDs, one for the original overlay and the other for the super-peer overlay and each super-peer maintains a routing table for each of these 2 overlays. A superpeer maintains an index in the form of a tuples of Client (C) and the key for the file present with C as fileId <fileId, C> over its clients data. These tuples are then published over the super-peer overlay by the super-peer (S) for other super-peers within the network using the following triple: <fileId, S, C>. For example as in Figure 7 where A, B, C, and D represent super-peers and a1, a2 and a3 represent A's client. Index of files f1, f2, and f3 is maintained in A and these files are stored at a1, a2 and a3 respectively. A then publishes these tuples <f1, a1>, <f2, a2> and <f3, a3> into the super-peer overlay network, in the form of <f1, A, a1>, <f2, A, a2> and <f3, A, a3> respectively.

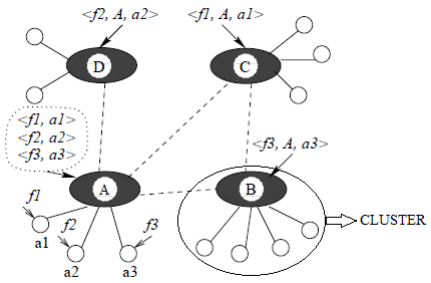


Figure 7: Illustration of a Super-Peer Overlay Network.[15]

As per current lookup algorithms, if there is a query Source and destination node present in the same cluster then this cannot be realized by the algorithm and the query messages gets routed across multiple systems before reaching the destination node. Among the hop systems there could be nodes present very far away (different country or continent) from each other, this could lead to high latency. Whereas in the super-peer based lookup algorithm, super-peer simply forwards the query to the destination within the cluster and avoid routing the query across multiple autonomous systems. This minimizes both network hops and latency as well as reduces network traffic.

If a new client joins/leaves or client insert/delete the file with key as fileId, then the interaction among the participants happen using the tuples and the triples for making the appropriate change in the routing table for the affected nodes. In case of super-peer, it is itself responsible for publishing/withdrawing its data indexes to/from the super-peer overlay. [15] discusses how these Super-peers is chosen:

- Significant processing power for routing large amount of overlay traffic on a overlay network of super-peers.
- High bandwidth and fast access to the internet e.g. gateway routers.
- High availability.
- It should have a large storage and memory capacity as they keep the index of files within its cluster and also the triples published by other super-peers.

Simulations in [15] shows that the superpeer lookup algorithm finds the files faster, consumes less processing power and bandwidth as compared to the existing location and routing algorithms. The results are compared in terms of Relative Hop Penalty (RHP) - ratio of the actual IP hops to route a query on the overlay network versus the ideal IP hops on the underlying IP network to the query destination node, Relative Delay Penalty (RDP) - ratio of the actual time to route a query on the overlay network versus the ideal network latency on the underlying IP network to the query destination node, Aggregate bandwidth Cost (ABC) - aggregate bandwidth consumed (in bytes) by each query, and Aggregate Processing Cost (APC) - aggregate processing power consumed (in CPU cycles) by each query.

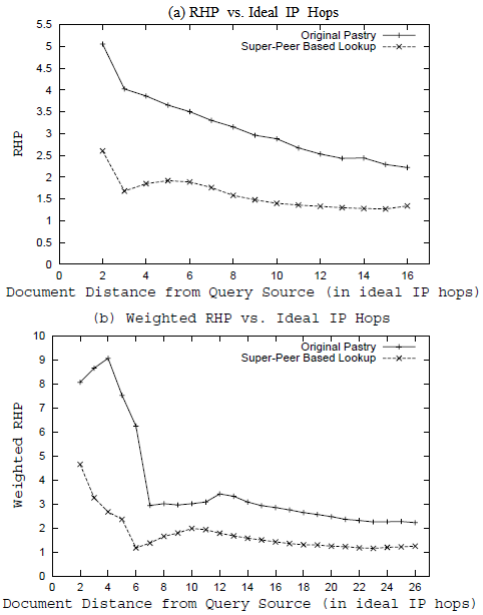


Figure 8: RHP vs Ideal IP Hops[15]

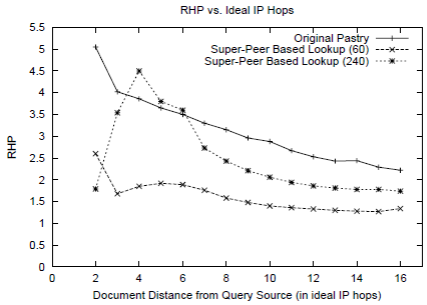


Figure 9: RHP vs. Ideal IP Hops. The RHP for the super-peer based lookup is measured under 60 and 240 super-peers respectively.[15]

In figure 8, The super-peer based lookup has much lower RHP than the original Pastry also the routing overhead gets reduced by about 34-81%. In Figure 9, there are 60 and 240 super-peers on the secondary overlay network. Here the one with less super-peers on the secondary overlay reduces the routing overhead on the secondary overlay for locating super-peers for query destinations this in turn improves the performance of the algorithm as well. RDP gets improve by about 10-43% (Figure 10) and ABC and APC gets reduced by about 40-50%.

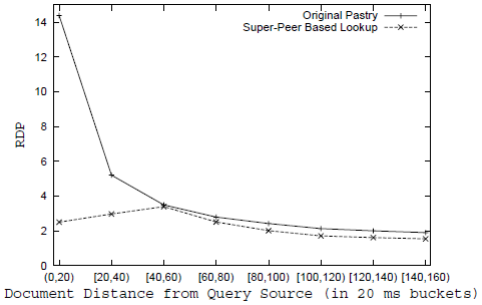


Figure 10: RDP vs Ideal Latency[15]

There have been several proposals for making Pastry an efficient P2P system which will be compatible for heterogeneous system as well. [16] focuses on some issues with Pastry and heterogeneity and proposed a **Statistic model** in order for Pastry to work in a heterogeneous environment. The heterogeneity can be among participants (different packet forwarding capacities, network connections, availability) and also in the network (internet, MANET or mobile network) and by not taking heterogeneity into consideration following issues can arise:

- **Reduced routing performance and increased maintenance cost**. Pastry follows a greedy approach for the routing scheme i.e. it only considers the numerical distance. But in heterogeneous environment there can be factors which will have significant impact on the performance and not considering those could decrease its performance.
- **Load is not actually balanced.** Pastry distributes workload equally among peers. But in heterogeneous environment, there can be nodes with high resources (like memory/ storage) and there can be nodes with low resources as well. In this case the node with low resource could get overloaded where as the powerful node with high resources could get under-load. And if large number of weak nodes gets overloaded, they will start rejecting the queries and this could also result in resigning from the overlay and finally will increase the maintenance cost as well.
- **Open to attacks**. There is no guarantee for all the nodes to be trusted. Pastry has few protections from adversary nodes. Sybil Attack or Eclipse Attack from small number of adversary nodes will undermine the overlay.

Based on resources (like computation power, battery power,memory capacity, etc.) of a peer, the Statistic Model was proposed. It suggested that the performance of an overlay network is impacted based upon three factors: node type, node health and link quality as shown in figure 11. Node types are of three kinds: weak node, normal node and strong node. Based on computational power, battery type, available storage/ memory, and connection speed a node is charaterized.

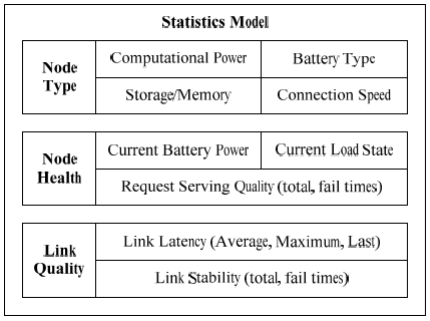


Figure 11: Maintaining procedure of statistics model.[16]

Node Health (NH) is determined by Current Battery Power (CBP), Current Load State (CLS) and Request Service Quality (RSQ) - keeps track of Total Query Times (TQT) and Failed Query Times (TFT); and Link quality is derived from Link Latency (LL)- comprising of Average Link Latency (ALL), Maximum Link Latency (MLL) and Last Link Latency (LLL), and Link Stability (LS) comprising of TQT and Failed Times (FT).

- $NH = CBP \times CLS \times \frac{TQT}{TQT-TFT}$
- $LS = \frac{TQT-FT}{TQT}$
- $LQ = \frac{1}{ALL} \times \frac{TQT}{TQT-FT}$

Memory cost of this model is small and affordable, computing cost is also negligible and the bandwidth cost of the model gets deduced from Pastry System except for CBP and CLS which is quite affordable. The statistic model[16] as compared to the one which currently exists offers the following benefits:

- More efficient in routing using Adaptive Routing: The Adaptive Routing follows a greedy algorithm through the key space. This new routing algorithm enhances the basic approach used in Pastry by taking into consideration node health and link quality. Several proposals like Proximity Neighborhood Selection (PNS) have been made for improving Pastry and demonstrated that routing along the lowest-latency hop that makes progress in key space can reduce end-to-end latency, although their results were based on simulations where the per-hop processing cost is ignored. However, the impact of the link quality and node health should not be ignored. When the routing algorithm chooses a weak node (low battery, high churn rate), there is a high possibility of routing message getting lost. Similar things happen when the routing algorithm choose a node whose link quality to the current node is low (unstable, too long latency).

- Timely and accurate fault detection: Lookup in a DHT with routing can be achieved via several paths. So, the node has to know that the previous request failed in order to send the next request. If the timeout is too long then the requesting node might waste time, waiting for a response from the departed node thereby increasing the overall latency. If timeout is short, the contact node may not have received it yet; the request may be still in process or may be due to congestion the response is queued in the network. Here, if another request to the contact node is sent, it might cause further dropped packages. This will result in transmitting more requests. The requestor should know that the late response is an indication of node failure and not of processor overload or network congestion. And therefore choosing a timeout is a very delicate task. The statistics model gives a more accurate and reasonable calculation of timeout using the three parameters: average latency, max latency and last latency.

- Maintenance cost is less than Pastry: The current implementation (FreePastry) sends a heartbeat message to nodes in leaf-set and routing table at some fixed interval, to keep Pastry in consistent state. Having this interval too short to keep a close eye on the unstable nodes will increase the maintenance cost for highly stable nodes unnecessarily. Increasing the interval might decrease the routing performance, which may put Pastry in inconsistent state.

- Load Balance is also reasonable: As per the algorithm proposed in Statistical Model, more workload is given to the normal nodes and the strong nodes as compared to the weak nodes. Usually weak nodes have a lower NH and LQ and hence overloading a weak node is very rare. Somehow if a node gets overloaded then using the modified algorithm, it will receive no or less workload in the following time and can be easily recovered from this temporary overload. This makes the modified algorithm better than the original algorithm.

- More secure than Pastry: In Statistics model with Request serving quality, the modified routing algorithm can lead to a negative feedback cycle and greatly improve the routing quality, so attacks like Eclipse Attack can be avoided. When the requestor node receives several false routing information from an adversary node, Request Serving Quality will get reduced so low that LQ will drop dramatically, and then with high probability the adversary node will no longer be used by the requesting node. And since the adversary node cannot affect the system so the robustness of the overlay will also gets improved.

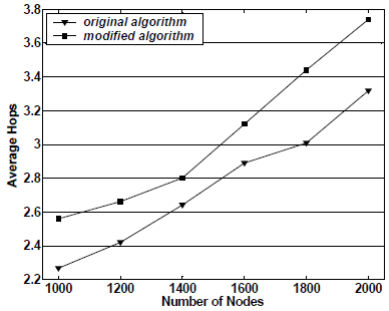


Figure 12: Average hops VS. number of nodes.[16]

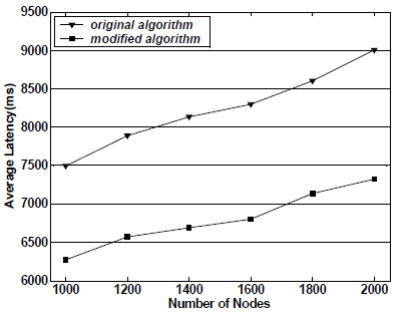


Figure 13: Average latency VS. number of nodes.[16]

In the simulation of [16], node number varies from 1000 to 2000. In Figure 12, it can be seen that there is increase in the average hops per message. But it is not too much. The average latency per message gets dropped by about 20% (Figure 13). If there are more nodes present in the overlay,

the average hops per message gets increased and the new routing algorithm will have to work more (because of the increased nodes) and hence the performance will also get improved.

5 Transformation of Java desktop application to support mobile applications

The existing Java applications for desktop would not run in the mobile environment without modifying the code. Although Java is said to be based on write-once-run-anywhere philosophy, but when it comes to mobile phone [17], it is not true because of the resource constraints like for example in a desktop, Java VM works as a bridge between desktop and OS; similar kind of VM is required in mobile phones as well for running the J2SE code in small computing devices like mobile phones. Each of the OS for mobile phones has its own SDK and requires specific programming skills for language as well as API. The following section will be covering two cross-compiling tools: XMLVM and XMob.

5.1 XMLVM [18]

XML-based programming language (XMLVM) exploit benefits of XML, it gets created automatically from Java class files [19]. The cross-language translations are accomplished by using the XSL-transformations of XMLVM programs. XMLVM [20] offers a flexible and extensible cross-compiler toolchain as shown in Figure 14.

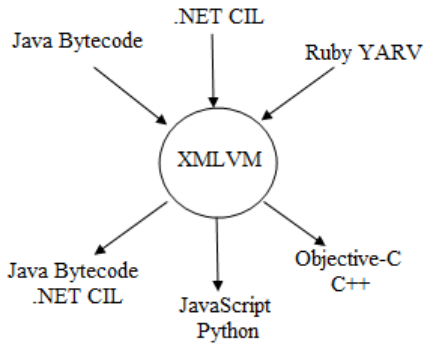


Figure 14: XMLVM as a blackbox..[20]

XMLVM API abstracts the details or transformations; it does so that the developer can implement the program in any high level languages such as Java or .net Framework. XMLVM uses the targeted output which is formed as a result of the compilation of the program. The output, Java byte code for Java language or .net CIL for .net framework is used by the XMLVM to perform transformations using the language semantics and syntax of the programming language used for writing the program to the targeted platform program, in our case which can be targeted to generate executable or program for the targeted platform. XMLVM achieves these through XML transformations through XSLT [21] on the intermediate XML generated by the XMLVM API. Figure 15 shows the XMLVM transformation process

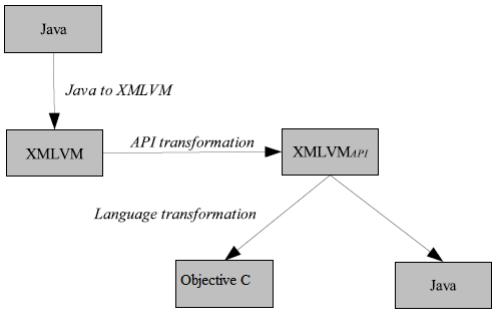


Figure 15: XMLVM transformation process.

[19] explains about how to translate Java code into another programming language. First the Java source code gets converted to XMLVM by a tool. After that the API transformation of XMLVM is done so that the API from the source can be adapted in the target language. The original code was written using some specific APIS like JUnit. So, if the program has to be translated to some mobile support language like J2ME, this API needs to be adapted to the appropriate API available for the target language. For each API there exists a XSL-stylesheet that can adapt the API. The result from the API transformation is still a XMLVM program which still uses the XMLVM notations like . for the scope operator. The simple renaming of operations are quite easily achieved by appropriate XSL-stylesheet, but for complex operations complex XSL-stylesheet is required. Each API transformations is handled by its own XSL-stylesheet and depending upon how many different libraries(e.g., JUnit, CORBA etc) are used, multiple stylesheets may be applied. The output from this transformation is still XMLVM program.The final step after generating XMLVM program from API transformation is to generate code for the target language. This is done using another XSL-stylesheet. Here the XMLVM instructions are mapped one by one to the target language without attempting to reverse engineer the original Java program. And as the Java VM is based on a simple stack-based machine the same can be replicated as a stack-machine in the target language. The intention of this approach is not to generate readable code but the correct code that uses the API of the target language. But using XMLVM has certain limitations as well:

- XML is not a very efficient way of expressing data.
- Mobile phones have light weight applications, this could mean that the libraries used in the Java code might not be available or supported for mobile phones. In this scenario the approach might fail.
- There can also be packages that have to be included in order to work on mobile phones. Those packages have

to be manually added after the target code gets generated. This can be detected while compiling the target program for bugs and errors.

- On slow wireless networks, before using XML as a data exchange format, have to do some real device testing. As nowadays latency is issue than transfer rate.

5.2 XMOB [22]

The authors of [22] compared a lot of tools like XMLVM, Neomades, etc. for transforming the source language like Java to some other target language. Table 1 shows the comparisons between those tools used for JAVA as a source programming language. Write for the source language and Run determines which target platform is supported for, Interpreter or Cross-compiler.

						Supported Platforms					
	Write	Run	Look'n'Feel	ystem Access	Hardware	iOS	Android	Blackberry	Windows Phone	Symbian	Bada
Tabris	Java	I	Yes	Yes	Yes	Yes	Yes	No	No	No	No
Neomades	Java	C	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
XMLVM	Java / .Net	C	Yes	Yes	Yes	Yes	Yes	No	No	No	No
Codename One	Java	C / I	Yes	Partial	Unknown	Yes	Yes	Yes	Yes	No	No

Table 1: Result from the survey on XMT in 2013.[22]

They also proposed another language dedicated for mobile development, XMOB. The basic objective of this tool is to capture sufficient common and high-level concepts that can be shared with different platforms and generate a native code skeleton, using the MDA approach i.e. through model refinement and the use of meta-modeling languages like MOF and UML2, standardized by OMG. XMOB has been designed in a modular way using MVC pattern. It consists of three sub-languages: **xmob-data**, deals with data sources and their access; **xmob-ui**, it only covers the visual elements. Theywill be placed and linked together automatically during compilation; and **xmob-event**, event handling is achieved using xmob-event i.e. links data elements and user interface together, for example button1 on Screen1 should be linked to Dialer then the xmob-event file will look like:

```
1  --User-defined event
2  on buttonpress[Screen1.button1] do {
3    open Dialer
4  }
5
6  --System-defined event
7  on networkavailable[SYSTEM] do {
8    close Dialer
9    open Hello
10   fetch cnnNews into Hello.lists[0]
11 }
```

Figure 16: Snippet for xmob-event.[22]

XMob consists of a cross-compiler as shown in figure 17. For implementing this solution Eclipse Modeling Framework (EMF) tooling is used for meta-modeling and modeling. XTEXT is used for creating text editor and concrete syntax for languages. The cross compiler of XMOB is based upon the MDA as shown in the figure 17.

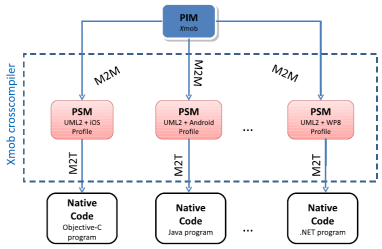


Figure 17: MDA-compliant architecture of the Xmob crosscompiler.[22]

Platform Independent Model (PIM): XMOB is independent from any OS so that the focus can be on the essentials rather than the technical details. Platform Specific Model (PSM): In this model the technical details of specific platforms will be taken into consideration. Using model-to-model (M2M) transformation (KERMETA), PIM can be converted to the specific PSM. From the PSM the code has to be generated, which can be achieved using model-to-text (M2T) transformation (XPAND).

6 Conclusion

Mobile Phones/smart phones are the epitome of the wireless technology and, Peer-to-peer networking is the future. This paper discusses various approaches, issues and challenges in this field of integrating P2P paradigm and mobile nodes/smart phones. Currently there is an explicit assumption of peers (like mobile nodes) being homogeneous in resources, i.e. the peers in a network with different capacities will be sharing the same load. For e.g. if there is a peer with less resources (week node) and another peer with very high resources (strong node) then the load will be equally distributed among the two and this could result in overloading the week node and under-loading the strong node, hence to overcome this issue approaches focusing on this heterogeneity in a network has also been discussed in the paper. In [15] Superpeer with heterogeneity capabilities has been discussed and in [16], a statistical model has been proposed for heterogeneous system.

At the end the paper discussed two cross-compiling tools, XMLVM and XMOB. XMLVM cross-complies Java program into other language as per requirement and what can be the trade-offs while trying to use Java desktop application in Mobile phones. And XMOB uses the MDA technique to transform the source language into another. There are still open question on using desktop applications in mobiles: These cross-compiling tools will be able to transform the Java code up to a certain limit. How to reduce memory consumption of the application for using the application efficiently in a mobile phone? In desktops, the users have keyboard and mouse to interact with the application how will this be handled while transformation. The modern devices like smartphones uses touch and positioning like advanced features. How will those be taken into consideration? XMob

uses meta-modeling concept for this purpose which can be applied to the new applications being developed. But, for the existing desktop application this tool is not successful as there is no meta-model for it.

Literatur

[1] W. Kellerer, "Dienstarchitekturen in der Telekommunikation - Evolution, Methoden und Vergleigch," in *Technical Report TUM-LKN-TR-9801*, 1998.

[2] K. Graffi and P. Wette, "Adding Capacity-Aware Storage Indirection to Homogeneous Distributed Hash Tables," in *Proc. of IEEE International Conference on Networked Systems*, 2013.

[3] F. Chowdhury and M. Kolberg, "A Survey of Peer-to-Peer Solutions in Mobie and Cellular Networks," in *Proc. of Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, 2012.

[4] W. Kellerer, G. Kunzmann, R. Schollmeier, and S. Zöls, "Structured peer-to peer systems for telecommunications and mobile environments," in *AEÜ - International Journal of Electronics and Communications*, 2006.

[5] I. Kelényi and J. K. Nurminen, "Optimizing Energy Consumption of Mobile Nodes in Heterogeneous Kademlia-Based Distributed Hash Tables," in *Proc. of International Conference on Next Generation Mobile Applications, Services and Technologies*, IEEE Computer Society, 2008.

[6] M. C. Castro, A. J. Kassler, C.-F. Chiasserini, C. Casetti, and I. Korpeoglu, "Peer-to-Peer Overlay in Mobile Ad-hoc Networks," in *Handbook of Peer-to-Peer Networking*, Springer, 2009.

[7] V. Darlagiannis, A. Mauthe, and R. Steinmetz, "Overlay Design Mechanisms for Heterogeneous, Large-Scale, Dynamic P2P Systems," in *Journal of Network and Systems Management*, Kluwer Academic Publishers-Plenum Publishers, 2004.

[8] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable Peer-To-Peer lookup service for internet applications," in *Proc. of ACM SIGCOMM Conference*, ACM, 2001.

[9] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proc. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Springer, 2001.

[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," in *Proc. of ACM SIGCOMM*, ACM, 2001.

[11] P. Francis, "Yoid: Extending the Multicast Internet Architecture." Unpublished Paper, available at http://www.aciri.org/yoid/, 2000.

[12] H. Sözer, M.Tekkalmaz, and I. Korpeoglu, "A peer-to-peer file search and download protocol for wireless ad-hoc networks," in *Journal of Computer Communications*, Elsevier Science Publishers B. V., 2009.

[13] Q. Hofstätter, S. Zöls, M. Michel, Z. Despotovic, and W. Kellerer, "Chordella A Hierarchical Peer-to-Peer Overlay Implementation for Heterogeneous, Mobile Environments," in *Proc. of International Conference on Peer-to-Peer Computing*, IEEE Computer Society, 2008.

[14] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proc. of Multimedia Computing and Networking*, SPIE, 2001.

[15] Y. Zhu, H. Wang, and Y. Hu, "A Super-Peer Based Lookup in Structured Peer-to-Peer Systems," in *Proc. of International Conference on Parallel and Distributed Computing Systems*, 2003.

[16] D. Zou, Y. Gan, and H. Qu, "A Statistic Model to Pastry Based on Heterogeneous Environment," in *Proc. of IEEE International Conference on Computer Science and Information Technology*, IEEE Computer Society, 2009.

[17] J. Keogh, "J2ME: The Complete Reference," McGraw-Hill/Osborne, 2003.

[18] A. Puder, "An XML-based Cross-Language Framework," in *On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops*, Springer, 2005.

[19] A. Puder and L. Wang, "Cross-Language Functional Testing for Middleware," in *Proc. of International Conference on Testing of Communicating Systems*, Springer, 2005.

[20] XMLVM. http://www.xmlvm.org/, 2012.

[21] W3C, "Xsl Transformations (XSLT)." http://www.w3.org/TR/1999/REC-xslt-19991116, November 1999.

[22] O. L. Goaer and S. Waltham, "Yet Another DSL for Cross-platforms Mobile Development," in *Proc. of the International Workshop on The Globalization of Domain Specific Languages*, ACM, 2013.

General attacks on p2p-based online social networks

Eshant Tulsidas Gour
Project Group - A Peer-to-Peer Framework for Social Networks
Universitat Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email - geshant@mail.upb.de

Abstract

Most of the current Online Social Networks(OSN) are logically centralised. This particular architecture uses a central server for storing user information and data and the control over the service belong to the central entity. Therefore, In order to share their data with friends, people are forced to share their data with the provider and consequently they loose control over it. There are certain drawbacks of Centralised Online Social Networks(COSN) including privacy, provider dependence, scalability, etc. In order to overcome these challenges, efforts are being made to decentralise Social Network Services(SNS). A common way of building a Decentralised Online Social Networks(DOSN) is by using the peer-to-peer(P2P) architecture.

The paper aims to identify the security goals of OSN. The paper also focuses on the current framework and also discusses the possible attacks on the P2P framework with their defense mechanisms.

1 Introduction

Nowadays, OSN is the current trend amongst the people. In-fact, today OSN are some of the most visited websites [2]. Through OSN people can find their old friends and beloved ones. They communicate with them by using OSN. Moreover, by sharing pictures and other additional personal information people remain up-to-the-minute with each other. As users give a lot of private and personal data to the networking sites while registering, therefore, they expect their information to be safe and secure from threats and attacks. But, OSN are open to many security threats and attacks such as hacking, Denial of Service(DoS), destruction of data, etc. This has remained the main concern among the users of the social media.

Most of the OSN sites use the client-server architecture for storing and maintaining the user data. They store and maintain the data on the company's servers. This increases the high security concern among the users as their data is highly vulnerable to data leakages and privacy breaches. This type of architecture is always open to single point of failure. A single breach could cut-off or corrupt the outgoing and incoming contents. Therefore, the security of the entire system is always under threat due to the malicious attacks or careless administration. Currently, users are not having the complete command over who can access their data. Most OSN provide settings, which the user can use to limit the number of viewers over his/her data. Significantly, there is no guaranteed protection by the provider of the service itself.

To protect the privacy of the users and to increase the security, it is important to hand over the control of the data to the user. This has led to the aim of developing a DOSN in which the profile owner himself have the ownership of his personal data. DOSN do not depend on centralised storage of data but it provides individual control over the data. With DOSN data can be stored anywhere for example, it can be stored on the personal computer of the profile owner. With the removal of central storage of data the fear of single point of failure is also gone. It may increase the availability and scalability of services and can improve privacy [16]. The paper focusses on defining and applying the security concepts that best fit in the field of DOSN.

The rest of the paper is structured as follows: Section 2 introduces the security threats arising from online social networks. Section 3 discusses the security goals of the online social networks. Section 4 shows the fundamentals and mechanisms used in the current P2P framework and give a brief description to each of the mechanism. Section 5 discusses some of the attacks on the P2P network with their defense mechanisms.

2 Security threats arising from online social networks

With the increased use of social media, the world is becoming a more smaller and interconnected place. Social media has a large user base and contains a large amount of information, which is more than enough to divert attackers. These attackers play an important role in posing threats to

the social media and these threats affect the user's privacy, identity and confidentiality. Most of the users are not aware of such threats and hence, it is very important to understand these risks and prevent the loss of user's personal or private information. Some of the important threats are discussed below.

- **Identity theft** - In Identity theft, attackers create false identity by stealing someone's personal or private information such as names, birth dates, addresses, credit-card numbers, etc. By using these information the attackers commit fraud with victim's identity. In OSN, attackers can acquire the identity of permitted users and accesses their profile, friends, communication history and so on. Profile cloning can be achieved through identity theft in which attackers create a profile of some user who already has a legitimate profile on the same network. This makes it easy for the attackers to initiate trust relationship with the victim's contact list by simply sending the friend requests.

- **Profile porting** - In profile porting the attackers uses the identity of some user present in some OSN. Now, they create a profile with same identity in different OSN, where the victim may or may not be present.

- **Malware, Viruses and Worms** - Attackers create Banner ads, fake links, fake profiles and multimedia content such as images, audios, videos, etc. to attract users. When users click on such links or content they are redirected to sites infected with malicious code. There are also some malicious codes which spread through pop-ups and they do not require a click to spread.

- **Spam** - Spam can be any kind of unsolicited online communication [12]. In OSN, spam mostly comes as news feed, wall posts and messages. Nowadays, Users spend most of their time on OSN rather than checking e-mails, therefore, this type of spam is more powerful and effective than the traditional e-mail spam. These spams usually contain links or advertisements which lead the users to harmful malware sites.

- **Information leakage** - Information leakage means the leakage of some kind personal, private, corporate, confidential, customer or company's information through OSN. Some employees release and risk their company's confidential corporate data through OSN, which either results in loss of the company or poses some kind of security threat to it.

- **Image retrieval and analysis** - Through image retrieval and analysis, attackers collect images or videos from OSN and then with the help of automated face recognition tools, they find the profile of displayed user which reveals more information about the users.

  Beside these attacks there are many threats existing in OSN such as Plain Impersonation [5], Profiling[5], Crawling and Harvesting[5], Communication tracking[28], group metamorphosis[5], ballot stuffing and defamation[5], censorship[6] and collusion attack[5].

These threats intend to have an effect on user's privacy and confidentiality. There are certain security approaches which are proposed in [1], some of which are Proxy-Based Real-Time Protection for SNS, Home-Network Social Application, Reputation Mechanism using SNSs and so on. These security mechanisms helps in ensuring privacy of the users. It is equally important to make each user aware of the security of his/her data by spreading the information about SN threats, security mechanisms, etc.

3 Security goals and Requirements

In order to protect our system from various kinds of threats and attacks, we have to fulfill certain requirements i.e. Security objectives. According to my view, the most challenging goals in using OSN is the privacy of users and maintaining the integrity of data because of the large amount of personal information, a user gives at the time of registering in such sites. Coping up with all the goals simultaneously is enormously difficult.

3.1 Goals

- **Privacy** - Privacy has remained the main area of concern among the OSN sites. Users create profiles by entering the large amount of personal information and share it with friends and often with many strangers. In addition to this, highly sensitive details are also provided such as contact details including photos of their personal life. In most cases this data is stored in the central servers where a small security loophole can leave user's data vulnerable. The easiness of joining any SNS and the lack of security measures in most sites make it easy for the third parties to access user's data without their knowledge. Now, as this information is available easily, there are great chances of posing threats like identity threat, online stalking, etc. This factor makes it important for the users to limit their information among their friends. Most of the OSN offers privacy settings with which users can restrict outside access and can allot access rights to their contacts only. With this the control remains at the user's hand and the user can decide to whom they want to disclose their information or who can access their profile.

  Therefore, access control is the mechanism responsible for the protection of information which the user

publishes. To be more safe, the user should maintain different information blocks in their profile and access restriction to each block should be done distinctly[5].

- **Integrity** - Data Integrity is one of the most important objective of OSN. It refers to conserving the accuracy and uniformity of data throughout it's lifecycle. Users generate their own content and profile and can make the changes whenever they want to. The main objective of data integrity is to prevent this data from unaccredited modifications[5]. There are several factors that are responsible for causing unauthorised modifications to stored data. In most cases, this data is stored in the central servers, as a result of which it can get corrupted due to hardware or software failure. Malicious intrusion are also the main cause of integrity violations. There have been a large number of attacks on the system in last few years. If the attacker gains administrator rights then he can cause damage to the system by modifying system utilities or adding Trojans, etc. which could well result in data loss. There can also be problems with integrity of user profiles that in turn can affect privacy objectives. As the process of creating a profile is easy in OSN, therefore, it is unsure whether there is a real person behind the registered profile. This problem can be limited or prevented by authentication processes to assure that there are real people behind the registered profiles[5]. Restricting access to data can also be handy in protecting integrity violations. While it is also equally important to prepare disaster recovery procedures for virus attacks or server failures.

  Users leave a large amount of personal and private information on the network, What if the users want to delete this information completely or partially? Presently, almost all OSNs fails to deliver guaranteed deletion and removal of user data. They keep this information on the server but they block the access of other users to it. But, there is a high possibility that this information can be passed to third parties due to attacks or on the demand of the authorities.

- **Availability** - Availability may refer to how reliable the data is when users try to access it and how much time it takes for any user to access that data. The user's profile and its content should remain available until it is deleted by its owner. The content should be available even if the owner of the profile is offline and it should also be available even in the face of attacks and faults. Whenever, any reader tries to access the profile, he/she should be able to get the most recent content rather than the old ones. In case of OSN sites that helps their users to find job positions[5], it is very important to keep users data available all the time. While utilising services in the decentralised environment it is useful to execute the cooperation among the nodes and prevent denial-of-service attacks which are the main trouble for data availability[5].

3.2 Requirements

- **Access Control** - Controlling access to the information is a challenging task for an ordinary internet user as it requires users to act as system administrators to specify and configure access policies for their profiles. Nowadays, most OSNs provides group based access control mechanisms with which the user can apply access policies on the different group of friends and strangers. The security goals remain similar in both user based access control and group based access control[10]. For every piece of information, document or profile page, user should be able to specify access policies for individuals as well as for groups. Each and every piece of information is known as SharedItems[10]. Users should be able to change the access rights policies whenever they want to change it. Access control mechanism tries to maintain the availability, confidentiality and integrity of the documents.

- **Registration and login** - The main security requirement for OSN is registration and login. Whenever new users want to use the service, they have to register first. After registering users would be allotted with a unique userID and authentication information which should be stored confidential with integrity[10]. The login feature allows the users to enter the userID and password which the user has authenticated at the time of registration phase. By doing so users get logged in and then can successfully use the service.

- **Secure Communication** - Secure communication is the most important problem to face in online social networks. Whenever a user tries to communicate with other user i.e. during instant messaging and live chats, all sent messages should directly go to the specified receiver[10]. For this communication process, both the sender and the receiver should be authenticated and integrity and confidentiality should be provided by the communication[1].

4 Framework

Since, the centralised architecture of OSN brings various drawbacks with it, therefore, several attempts have been made to decentralise the OSN by using P2P technology so that the users could store their data in the distributed network. To ensure the privacy and integrity of user's data in distributed network, the data has to be encrypted and a very good access control mechanism has to put in effect so that the outside access over user's data is prevented[9]. While, on the other hand it is also very important to make sure that

the data is securely stored and it is easily available on user's desire. Lastly, the communication between the users should be authenticated and secured[10].

This section will mainly focus on the security mechanisms used in the P2P framework up-till now. These designs and mechanisms were used in the previous project groups, that is, "A Distributed Framework for Social Networks", 2011[9] and "A Peer-to-Peer Framework for Social Networks", 2012[2].

4.1 Fundamentals

- **Pastry** - To route messages in P2P framework, Pastry has been used. In substantial networks, the most difficult and challenging task for a peer is to communicate and send messages to the other peers. Therefore, Pastry plays an important role in passing messages from one peer to the other. To route data successfully in the network Pastry uses **Distributed Hash Table (DHT)**, which in general provides lookup service. It is fully decentralised, scalable and reliable. Every single node in the Pastry has unique identifier which is generally known as nodeID. The nodeID is a value between 0 and 2^{128} -1[9].

  Whenever a node enters the network it gets a value assigned to it. Every node in the network keeps a little information of its neighbouring nodes in the network to minimise the distance a message travels. So, whenever a node is encountered with a message and a key, the node systematically routes the message to that node whose nodeID is numerically closer to the specified key. The presumed number of routing steps can be determined by O(log N), where N is the number of nodes in the network[20].

- **PAST** - It is the large scale storage system built on Pastry. In P2P network data is stored and distributed among distinct peers. Now the peers those are storing the data cannot go offline, they have to be online every time to provide data to the users. In case if a user wants to access the data and the responsible peer is offline, then the data availability is not achieved as the peer fails to deliver the data to the desired user. Therefore, to evade this condition data replication is used which is provided by PAST[9].

  PAST is formulated of nodes that are connected to the internet. Each node in PAST is efficient i.e. each node can establish client requests to place or retrieve files. The original data is copied on a few other peers for achieving availability. Each node in PAST is assigned with a 128-bit nodeID(unique identifier) which is generally obtained by hashing the public key of the node. Moreover, each and every file that is added to the network is assigned with a 160-bit fileID. Here, fileID is obtained by hashing the file name, it's owner's public key and random salt[19].

  Whenever any file is inserted in PAST, it is always routed to the k nodes whose nodeID's are closest to the fileID. Then the copy of the file is stored by each of these nodes. In case of replication a lookup request is placed towards a node with nodeID closest to the requested fileID.

- **DHT**[26] - The main functionality of DHT is to provide a lookup service. A DHT distributes the data over various nodes and use routing to lookup the node for specified data. The storage mechanism of DHT is similar to that of a hash table and both provide similar type of lookup service. For storage, the put function is used for storing the data and it consists of the identifier and data which is need to be stored. Here, identifier can be referred as key and the data can be referred as value i.e. (key,value) pair. DHT store large number of (key,value) pairs. The get function of DHT retrieves the data related with the identifier. Routing is the basic function of DHT and it is the routing mechanism that supports the lookup function. It helps in delivering messages to the responsible node. Whenever a message enter the network it first queries any random node then that node forwards the message to other nodes. The nodes keep on forwarding the message to their neighbouring nodes unless and until the message is delivered to its responsible target node.

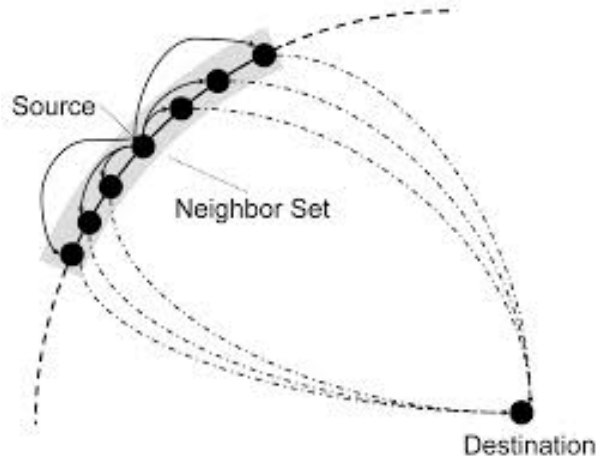


Figure 1. Depiction of Routing mechanism (Source: [11])

4.2 Cryptography

It is a technique of storing and transferring data in a secret manner. This is done by encrypting the information in impenetrable form. The main aim of cryptography is to avoid unauthorised access to data. Once data is encrypted it cannot be read unless and until it is decrypted. The component which is used for encryption and decryption of data is known as "Key". It also aims to deliver authenticity, integrity, confidentiality and non-repudiation. It ensures that only the authorised users can read the content and it also verifies the author of the content.

In the last Project Groups(PG)[9][2], cryptography is used for secure communication. In the PG mainly two types of cryptography are used namely Symmetric Cryptography and Asymmetric Cryptography.

4.2.1 Symmetric Cryptography

Symmetric cryptography is also known as "Secret key cryptography". In Symmetric cryptography same key is used for encrypting and decrypting information. This key is known to both sender and the receiver. This key should be kept secret from unauthorised users and invaders, otherwise they can easily decrypt the message if they know the key. Therefore, the security in secret key cryptography always depends on the user's ability to protect the key. This method is much faster than asymmetric cryptography. This method can be used in secure communication.

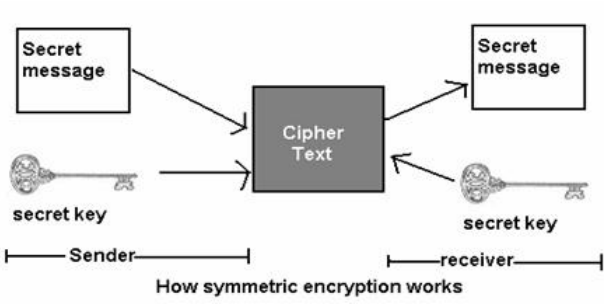


Figure 2. Symmetric Cryptography (Source: [8])

4.2.2 Asymmetric Cryptography

Asymmetric cryptography is also known as public key cryptography. In asymmetric cryptography, both sender and receiver possess two different keys. One key is responsible for encrypting the message while the other one decrypts the message. One of the key is known as public key which is generally known to everyone whereas the private key is known to its authorised user only.

Now, Suppose if user1 wants to send a message to user2. Therefore for sending the message encryption has to be done. Consequently, user1 performs encryption of the message by using the public key of desired recipient i.e. user2 and he sends the message to user2. Now, after receiving the message user2 can decrypt it with his private key. Now, what if user2 wants to reply back to user1? In this case user2 encrypts the desired message with the public key of the user1 and when user1 gets the message, he can decrypt it by using his private key.

This mechanism ensures the authenticity of communication. It also delivers confidentiality and non-repudiation. In the last PG[9], asymmetric cryptography is used in encryption and decryption of data and also in signing and verification of data.

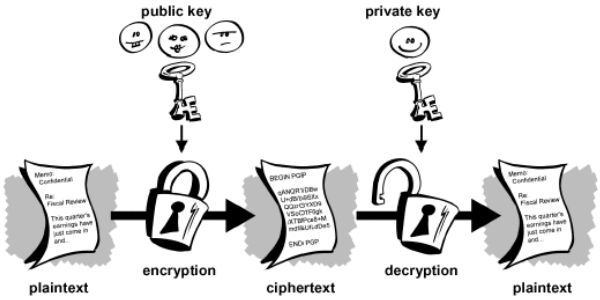


Figure 3. Asymmetric Cryptography (Source: [22])

4.3 Secure messaging

The main aim of secure communication is to deliver privacy and confidentiality. In other words we can refer it as to conceal the contents of any message from the prohibited users. However, it is also very important that no-one is able to modify the message when the sender sends the message to the receiver. The basic requirement for any OSN is communication. Every network node should be able to pass messages to other nodes and it should also be able to receive messages from other nodes. To achieve secure communication, it is very important to ensure only the permitted users or groups are able to see the messages[9]. While, on the other hand it is also important to demonstrate the genuineness of the message's source.

In the current P2P framework [9], two security mechanisms have been used for ensuring the secure messaging.

4.3.1 Encryption

It is the process of encoding some piece of information in an unreadable form, so that the information is safe from outside access. This mechanism makes sure that only the

permitted users see the messages and it denies the access to the unauthorised users. This can be done by encrypting the message with a symmetric key[9]. Now, the key has to be with all the users so that they can correctly decrypt the message and read it. The message can also be encrypted with public section of asymmetric key. Now, in this case no-one can decrypt the message except the receiver, because to decrypt the message private key of the receiver is required.

4.3.2 Signature

This mechanism is mainly used to demonstrate the genuineness of the source of the message[9]. This helps in achieving the authenticity and integrity of the sent messages. Here the message is signed with the private section of the asymmetric key of the sender. As the public part of the same key is nothing but the nodeID, therefore, it is available openly. This enables the recipient to verify whether the message was posted by the specified sender or not.

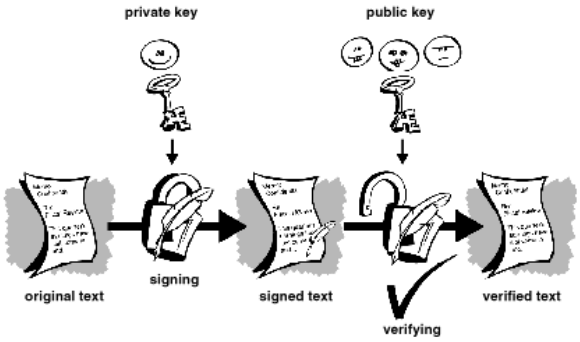


Figure 4. Signature Mechanism (Source: [22])

4.4 Data storage and retrieval

The storage and retrieval of data is managed by DHT which is its main functionality. DHT distributes data among several nodes and if it want to retrieves the data it executes routing to lookup for the node which is responsible for storing the data. Every DHT node keeps a little information about its neighbouring nodes in the network. DHT maps data with the nodes in the address spaces. DHT allots some part of the network to every node which is responsible for storing the data. Every node in the address space is assigned with a unique number i.e. identifier which helps in the lookup function. Now, every data object that arrives in the network gets an unique identifier ID. Wherever the data is stored in the network it can be identified with this identifier. Now the put function is responsible for storing the data in the form of (key,value) pair, where key is the identifier and value is the data object. Similarly, the get function is used to retrieve the data associated with the corresponding identifier[26].

4.5 Access Rights

This mechanism maintains the confidentiality and integrity of any document or an object in the network. Through access control policies users can restrict outside access over their data. A user should always be able to write access policies, read them and modify them whenever he wants to. But at the same time he should also have the access permissions to perform these operations. In the last PG Access Control List(ACL)[10] has been used as it allows a fine-grained access control over the data. Using ACL users can restrict individual access as well as the group access.

4.5.1 ACL

Access Control List is the list of authorised permissions which is generally attached to an object. The lists contains the user permissions for the associated object. The list states whether which user have the permission to access the objects. It also specifies what operations a user can perform on those objects.

If a request is made to access any object then with the help of this list, the list's owner can verify if the petitioner have the access permissions or not. Every information i.e. object is encrypted to evade illegal access. Objects always stores the required ACL with them. This ACL contains the necessary details that are needed to decrypt the object. The permitted users of this object have the necessary key for decrypting the object. Here the decryption key is encrypted with the public key of the permitted users. Now the decryption of the object would require the private key of the users. In this way it ensures that only the authorised users can access the information.

4.5.2 Read and Write access

The following description is based on the publication of Mr. Graf and his team mates which is available in[10].

Whenever a user wants to store new SharedItem, he signs it so that later he can verify the author of the item. A user should be able to define the read permissions for this item. i.e. which other users can read this item. For defining read permissions for a set of users, user creates a symmetric key. Then the user encrypts the SharedItem along with its signature, with this key. Then the user encrypts this symmetric key with the public keys of the allowed users. For n number of users, n number of encrypted copies of the key is created. These encrypted copies of the key are attached to the SharedItem. And this SharedItem is stored as the CryptedItem. CryptedItem is the SharedItem with key list signed by the user. Each stored item has ObjectID. This ObjectID helps in finding the location of the CryptedItem in the network. StoredItem can be retrieved with their respected ObjectID's. Whenever, a user wants to change the

already stored object, first he have to retrieve that object from the network. Since, the objectId is created by hashing the username therefore it contains the name of the user. The privileged users of any SharedItem can be changed by just changing the keys that are attached to this SharedItem. The item can always be retrieved by any node in the network, but the item can only be decrypted by those nodes which are there in key list. Every node in the network have its own ID. Now, if the ID of the retrieving node matches with any of the entries in the key list, the symmetric key is unpacked and the item is decrypted.

4.5.3 Group-based access control

First of all, a group can be seen as list of members and the members with special privileges i.e. administrators. Every user is allowed to create a group and the user who creates the group, he automatically becomes the administrator of the group. Only the administrators of the group can modify objects of the group and rest of the members just enjoys the membership of the group. When a administrator creates a group, he encrypts the symmetric key with the public keys of the group members[10]. Whenever a new member joins the group, administrator encrypts the copy of the symmetric with the corresponding public key of the user. To avoid outside access to the list administrator signs the list with his private key. A group can have one or more administrators. To prevent unauthorised access and illegal modifications, it is really very important that the administrators of the group have a same public/private key pair. For this, each group corresponds to the corresponding administrator group. This administrator group contains all administrators as members and this group gives the keys that helps in storing the group securely. The same keys are also used for storing the administrator's group securely. This ensures only the administrators can make changes in the group and only the administrator can add or remove a administrator.

5 Attacks and Defenses

The security problems in P2P networks are very much important and challenging. P2P system is highly open to various kind of malicious attacks such as DoS and DDoS attacks, Sybil attacks, Eclipse attacks, etc. In the following, we will discuss the possible attacks and their defense mechanisms.

5.1 DoS and DDoS attacks

Denial-of-service(DoS) and Distributed Denial-of-service(DDoS) attacks are getting more and more popular with the increased use of internet. These attacks are responsible for affecting the normal functioning of any service or it may even result in the loss of service. DoS attacks are performed by consuming the resources of the targeted computer. As a result, The targeted computer fails to provide the services to the users. It also affects the communication between the service provider and the user. DoS attacks overloads the targeted machine with external requests so much that it stops responding to the traffic. Such attacks are the root cause of server overload. In DDoS attack, the attacker capitalise on a large number of hosts and simultaneously sends the packets to the targeted hosts[15]. By doing so, the resources at the targeted hosts are consumed and it fails to provide or receive services. P2P networks are composed by large number and anonymous concurrently running hosts. Thus, one or more malicious nodes in the network can easily perform DoS or DDoS attacks to a victim server or client. [24]

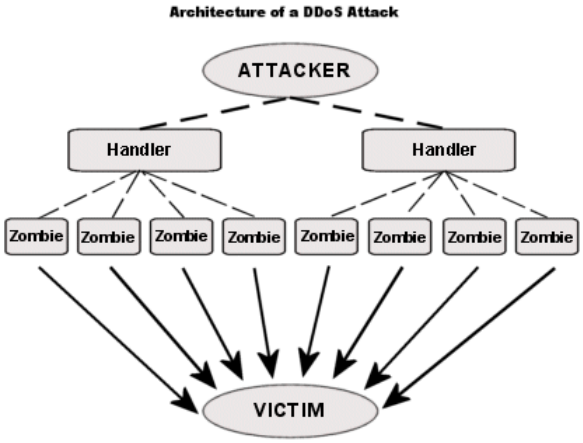


Figure 5. DDoS Attack (Source: [25])

Figure 5 shows the architecture of DDoS attack. The figure depicts that the army of the attacker consists of handlers i.e. master zombies and zombies i.e. slave zombies. The hosts of both the classifications are the machines that are infected by malicious code. The attacker handles and coordinates the attack through handlers which in turn coordinate and handle the slaves. The attacker sends attack orders to handlers and activates the attack process and these handlers in turn sends the attack orders to slaves ordering them to start the DDoS attack against a victim. As soon as the attack is triggered, zombies start to send large packets to the victim which floods the victim's network and consumes its resources.

The most important challenge in defending the DoS attack is detecting such type of attacks first. The heavy utilisation of the machine and the attacker's ability to involve vast number of machines in the attack makes it difficult to detect the source of the attack. The most broadly used technique against DoS attacks is "pricing" [17]. In pricing, the host gives puzzles to its clients before processing the client's request. By solving the puzzle the attacker will himself will go through the expensive computation. Every time the attacker tries to flood the host, the host allots him a new puz-

zle. This makes it difficult to establish a successful DoS attack. This method is not very effective against more distributed attacks.

5.1.1 TCP Syn Flooding Attack

It is a type of DoS and DDoS attacks. In this attack, an attacker sends a series of SYN(Synchronisation) request to the target system. By doing so, the resources at the target system are consumed and the system stops responding to the traffic. For initiating a connection TCP uses a process which is known as three-way handshake. First, the client sends a SYN request to the server to establish the connection. Then the server acknowledges the client by sending a SYN-ACK(Synchronisation-acknowledgement) and lastly client again sends an ACK(acknowledgement) to the server. By this the three-way handshake is met and connection between the client and server is established.

In TCP Syn flooding attack, the attacker ideally uses the fake IP address and then sends a SYN request to the target computer. As soon as the target computer gets the SYN request, it responds with a SYN-ACK message. Since the IP address of the attacker is fake, therefore, the victim node will not receive any ACK message from the expected client. Now, the attackers continuously sends a chain of SYN requests to the victim which consumes the resources of the target and it fails to perform the service.

One of the most used and effective method against this attack is Syn cookie[24]. Syn cookies eradicates the resources allocated on the target host. When a client sends a SYN request to the host, the host uses SYN cookies to send a SYN-ACK to the client, but it removes the SYN entry for that client. This enables the host to function properly as though it has never received a SYN. As soon as this SYN-ACK with SYN cookies is received by the client, the equivalent ACK with data regarding the original SYN-ACK is sent back to the host. Then the host uses this cookies and ACK to reestablish the original SYN-ACK and the entry for that original request. Once this is done the connection between the client and host is established. This entire process has avoided the SYN flooded queue.

5.2 Rational attacks

A notable challenge in P2P networks is the problem of co-ordination between the nodes. P2P system relies on the cooperation of nodes. In most cases, a selfish node will try to make the best use of system resources by utilising them and on the other hand it will try to minimise it's own resource sharing. This type of node is known as "Rational node". If the large number of nodes become selfish and rejects to cooperate, then the system may loose its stability.

Some P2P system tries to solve the problem of rational nodes. Just for an instance, Napster tried to solve the problem by giving users the title for the level at which they shared[23].

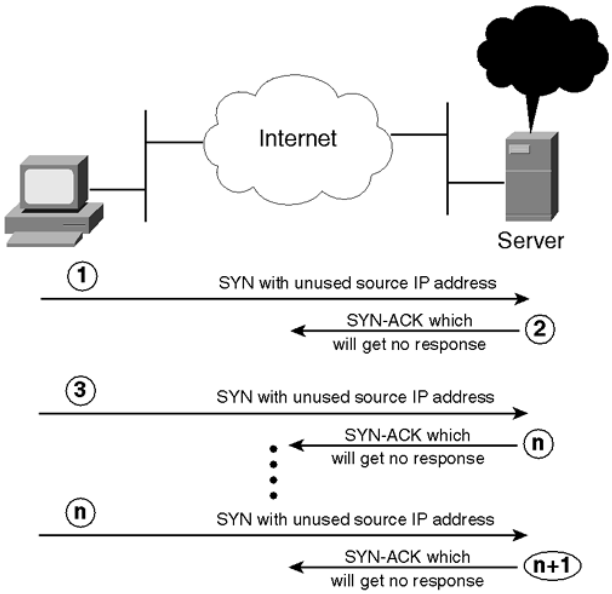


Figure 6. TCP Syn Flooding Attack (Source: [3])

While, Bit Torrent trade for blocks of data. In Bit Torrent, the more a node share with others, the more it will get back. So, more a node is willing to upload to others, the faster download it gets[23].

5.3 Poisoning attacks

Poisoning attacks are some of the major attacks in P2P networks. The main aim of poisoning attack is to break the integrity of the P2P systems. Attackers uses bogus file indexes, bogus routing tables, bogus IP addresses, etc. to achieve poisoning attacks.

5.3.1 Index poisoning attack

The main aim of this attack is to make searching more problematic for the users. P2P file sharing systems have indexes which users search to find locations of desired titles [14]. As index help the users in locating the desired content, therefore, the attacker targets the index of the data. The attacker adds a large amount of invalid information or records in to the index, which makes it difficult for the users to locate the desired content. There are several ways of poisoning the index of the file, for e.g. by using fake IP addresses, port numbers, randomly chosen identifiers, etc. One of the most common way of poisoning is by using the randomly chosen identifiers which do not match with any current file in the system. Now, when the users try to download a file, they fail to locate the desired file. The large amount of bogus

information makes it difficult for the server to establish a connection. Now, the users will again try to download a file with different identifiers, which may again fail. This goes on and users waste time by continuously trying to establish the connection. This sort of actions increases the average time spent in downloading any file. If the users does not succeed in locating the files after so many attempts, then it is possible that they may stop the search. Index poisoning attacks requires less efforts than other type of attacks because it requires less bandwidth and server resources. The attacker also does not need to transfer any file, but, simply have to add the invalid information to the indexes.

Almost all, P2P file-sharing systems are completely open to index poisoning attacks. It is formidable challenge to protect against the poisoning attack as the false records or information only had to be placed at one node. It can be limited by rating the sources of the files. If the sources are good and the file which they upload is also good and useful, the related peer will be given a high rating scores. If the sources are bad and the file is not useful i.e. it is being uploaded for poisoning the system, then the related peer will be blacklisted[14].

5.3.2 Routing table poisoning attack

The main aim of this attack is to poison the routing table and to make some malicious modifications in it. The routing table is constituted of peers and their neighbouring peers. Every single entry in the list consists of port number, neighbour's ID and IP address. Whenever, any peer joins the system, a routing table is generated and as other peers gets added or removed the table gets updated. In this attack, the attacker dodges the peer and forces the peer to add forged neighbours in to their routing table. The IP address of the forged neighbour is the IP address of the victim. Now, it is possible to poison the peer by just sending the bogus messages that clarifies different peers that victim's IP address does exist. After, receiving this message the peer may accept the victim node and include it as their neighbour. Afterwards, the poisoned peer may select any bogus neighbour from the routing table and can forward messages to it. This can lead to poisoning of more peers in the table which in turn could add more forged entries with IP address of the victim. Due to this the victim will receive large number of messages coming from the different sources.

The routing table poisoning attack is not as dangerous as index poisoning attack. Most P2P systems has the ability to automatically update the routing table. Therefore, When the peers realise that their neighbour does not reply with a valid message, the peers remove their neighbours from the routing table [15].

5.4 Routing attacks in Distributed Hash Table(DHT) Based p2p

As the routing process is one of the most important concept in P2P systems, therefore, it is very important to address threats against this process. In DHT based P2P network, each and every node is responsible for keeping and maintaining a routing table. The main functionality of the routing table is to assure the lookup and mapping of the keys. A single malicious node could dispatch lookups to erroneous and non-existent node. It can also corrupt the routing entries of other nodes.

These type of attacks are easy to detect because of the properties of DHT. With each hop the lookup gets closer to the key identifier, if the lookup fails to follow this pattern, then the querier(query node) detects the attack. Querier then finds the last good hop or the correct node and then asks for an alternative route[21]. However, to perform this check, each and every step of the process should be visible to the querier.

5.5 Sybil attack [7]

It is the most common type of attack in P2P networks. The main aim of this attack is to gain control over certain part of the network or the whole network. P2P networks are logical or virtual overlay networks built on an underlying network. Therefore, for each and every entity in underlay network, there is an equivalent identity in overlay network[24]. This implies that each entity in the underlay network is mapped to exactly one identity in the overlay network, in other words they maintain a one-to-one relationship. However, a malicious node is capable of demolishing the one-to-one relationship between the entities and the identities. This is achieved, when a single malicious entity represents multiple identities in the network. Therefore, attackers try to forge identities in the P2P networks to achieve their goal. Consequently, one entity gains the control over certain part of the network or the whole network. This is known as Sybil attack. The Sybil attack is also very common in ad-hoc and sensor networks. Reputation system is the important part of P2P networks. It is responsible for opinions, ratings, service providers and everything else on the P2P networks. It tells the users whether the entity is useful or it should be avoided. They are very common target of Sybil attacks [13]. Attackers extend the score of such entities which makes them look as if they are useful and users gets attracted to such entities and downloads or uses them. Attackers can also make large number of entities and can also increase their reputation by voting on them. This makes other users to believe that the entity is useful, and this allows the hackers to be more effect full.

Figure 7 depicts the Sybil attack. The figure represents three different types of nodes i.e. normal nodes, sybil nodes and a compromised node. The normal node makes the

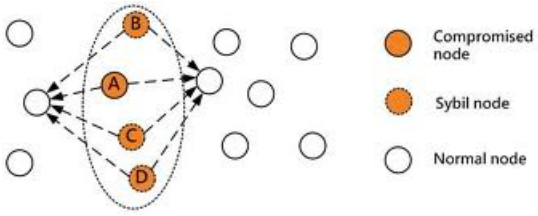


Figure 7. Sybil Attack(Source: [27])

unnecessary connection by selecting different entities i.e. node A, node B, node C and node D, which have different IDs. Here node B, node C and node D are the malicious nodes that are created by the attacker. This way an entity can gain a control over certain part of the network.

The prevention of Sybil attacks can be achieved through validation techniques[13]. A trusted central authority can be used to ensure a one-to-one relationship between an entity and an identity. The validation of identities can be done in two different ways, directly or indirectly. In direct validation, an entity asks the central authority to validate the identities. While, in indirect validation, an entity depends on the accepted identities which in turn confirms the validation.

To limit the Sybil attack, one important technique could be "pricing". Whenever, any node tries to join the network, other nodes offer it a cryptographic puzzle. Significantly, attackers have to solve the puzzles continuously in order to add more nodes to the network. One possible way to limit this attack could be assigning some kind of expiration time to the nodeID. Now, the attacker does not have enough time to generate nodes and execute the attack, as the node which has joined first starts to become invalid.

5.6 Eclipse attack

The Eclipse attack is the more general attack than the Sybil attack. It allows the attacker to control the significant part of the P2P network and it is performed by attacking the overlay maintenance operations. An Eclipse attack can be the prolongation of a Sybil attack, attackers achieve this by generating large number of identifications and this identifications act as a neighbour of a valid node[23]. The main aim of this attack is to divide the whole network in two or more partitions known as subnetworks. In case if the attacker wants to initiate the eclipse attack, he should have a control over certain nodes. After gaining control over the nodes, the attacker can divide the network in subnetworks. Now, in case if a node of one subnetwork wants to communicate with the node of another subnetwork, in other words, if the nodes want to exchange messages between them, at some point of time the message will pass through the malicious node of the attacker. The overlay node fails in forwarding the correct message and the network cannot be managed properly. Fraud and modification of messages can be prevented by using public key cryptography and digital signatures. "Pricing" could be the good technique to use against the Eclipse attack which is based on a Sybil attack. We can also use randomisation algorithms that controls the location of the nodes. Due to the random distribution of nodes, attacker is unable to control his node, as there are no strategic position decided. This makes it impossible to a network into subnetworks. Some other techniques that are handy against such type of attacks are verified routing table [4] and induced churn method [18].

6 Conclusion

Current OSN are centralised which brings certain drawbacks with it. These OSN stores user's data on central servers which is a high security concern as this type of architecture is always open to data leakages and privacy breaches. Moreover, users have to depend on providers for sharing information and communicating with their friends which is also an area of concern.Furthermore, the servers are always open to single point of failure, even a single breach or attack could cut-off the incoming and outgoing contents. Therefore to improve security and privacy of users DOSN have been adapted which gives the control of data to users themselves and eradicates the provider dependence.

This paper discusses that decentralisation is always a better option than current centralised online social networks. Moreover the paper discusses the general security goals and requirements of OSN. The paper shows the importance of privacy, integrity, availability, registration process, etc. to improve the security. Besides these how certain mechanisms can be used to increase the security of OSN has also been discussed. It shows that how the cryptography and access control mechanisms can be used to ensure the privacy, integrity and availability of data in distributed network. Symmetric and Asymmetric cryptography ensures that the user's data is stored safely while on the other hand access control mechanisms ensures that there is no unauthorised access over user's data. Additionally how encryption and signature techniques can be used to achieve secure communication with complete private and confidentiality has also been discussed.

The paper also focuses on security threats arising from OSN. Threats affects users privacy and identity, therefore it is necessary to prevent these threats. P2P systems are also open to various kinds of malicious attacks. Therefore, the paper addresses some of the general attacks that are practised on P2P networks and also discusses some preventive measures and defensive mechanisms against these attacks.

References

[1] R. Ajami, N. Ramadan, N. Mohamed, and J. Al-Jaroodi. Security Challenges and Approaches in Online Social Networks: A Survey. In *International Journal of Computer Science and Network Security(IJCSNS)*, 2011.

[2] P. Alavi, L. Bremer, D. Veena, J. Janiuk, L. Kangping, S. Gurumurthy, A. Luqman, A. Ma?cker, Y. Mexin, A. Morast, B. Jazayeri, C. Parekh, A. Petkau, J. Sander, and H. Zhaoi. Project Group 2012, A Peer-to-Peer Framework for Social Networks. In *Universitat Paderborn Fakultat fur Elektrotechnik, Informatik und Mathematik*, 2013.

[3] "Attacks". Common Network Attacks. `http://book.soundonair.ru/cisco/ch14lev1sec4.html`. Accessed June 13, 2013.

[4] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach. Secure Routing for Structured Peer-to-Peer Overlay Networks. In *ACM SIGOPS Operating Systems Review(OSR)*. ACM, 2002.

[5] L. Cutillo, M. Manulis, and T. Strufe. Security and Privacy in Online Social Networks. In *Handbook of Social Network Technologies and Applications*. Springer, 2010.

[6] A. M. G. de Paula. Security Aspects and Future Trends of Social Networks. In *Proceedings of The International Conference on Forensic Computer Science (ICOFCS)*. IEEE, 2009.

[7] J. Dinger and H. Hartenstein. Defending the Sybil Attack in P2P Networks: Taxonomy, Challenges, and a Proposal for Self-Registration. In *The International Conference on Availability, Reliability and Security(ICARS)*. IEEE, 2006.

[8] "EncryptDecryptdata". How to Encrypt/Decrypt Data in ASP.NET Using Symmetric Algorithm. `http://aspdotnetcodebook.blogspot.de/2011/07/how-to-encryptdecrypt-data-in-aspnet.html`. Accessed May 30, 2013.

[9] K. Graffi. Project Group 2011 A Distributed Framework for Social Networks. In *Universitat Paderborn Fakultat fur Elektrotechnik, Informatik und Mathematik*, 2012.

[10] K. Graffi, P. Mukherjee, B. Menges, D.Hartung, A. kovacevic, and R. Steinmetz. Practical Security in P2P-based Social Networks. In *The Annual IEEE Conference on Local Computer Networks(LCN)*. IEEE, 2009.

[11] C. Harvesf and D. Blough. Replica Placement for Route Diversity in Tree-Based Routing Distributed Hash Tables. In *IEEE Transactions on Dependable and Secure Computing(DSC)*. IEEE, 2011.

[12] G. Hogben. Security Issues and Recommendations for Online Social Networks. In *ENISA Position Paper*, 2007.

[13] B. Levine, C. Shields, and N. Margolin. A Survey of Solutions to the Sybil Attack. In *University of Massachusetts*, 2006.

[14] J. Liang, N. Naoumov, and K. Ross. The Index Poisoning Attack in P2P File Sharing Systems. In *IEEE International Conference on Computer Communications(ICCC)*. IEEE, 2006.

[15] N. Naoumov and K. Ross. Exploiting P2P Systems for DDoS Attacks. In *Proceedings of the International Conference on Scalable information systems(ICSIC)*. ACM, 2006.

[16] T. Paul, B. Greschbach, S. Buchegger, and T. Strufe. Exploring Decentralization Dimensions of Social Networking Services: Adversaries and Availability. In *Proceedings of the ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research(HTISNR)*. ACM, 2012.

[17] B. Pretre. Attacks on Peer-to-Peer Networks. In *Dept. of Computer Science, Swiss Federal Institute of Technology (ETH), Autumn*, 2005.

[18] K. Puttaswamy, B. Zhao, and H. Zheng. Securing Structured Overlays Against Identity Attacks. In *IEEE Transactions on Parallel and Distributed Systems(PDS)*. IEEE, 2009.

[19] A. Rowstron and P. Druschel. PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility. In *Proceedings of the Workshop on Hot Topics in Operating Systems (HTOS)*. IEEE, 2001.

[20] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the International Conference on Distributed Systems Platforms (DSP)*. Springer, 2001.

[21] E. Sit and R. Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. In *Workshop on Peer-to-Peer Systems(P2P)*. Springer, 2002.

[22] A. Toponce". Encryption And GPG Applications. `http://aarontoponce.org/presents/gpg/gpg.html`. Accessed May 30, 2013.

[23] Z. Trifa and M. Khemakhem. Taxonomy of Structured P2P Overlay Networks Security Attacks. In *World Academy of Science and Engineering and Technology(WASET)*, 2012.

[24] L. Wang. Attacks Against Peer-to-Peer Networks and Countermeasures. In *Helsinki University of Technology*, 2006.

[25] "Webmaster". What are DoS and DDoS Attacks? `http://www.techwarzone.com/what-are-dos-and-ddos-attacks/`. Accessed June 13, 2013.

[26] K. Wehrle, S. Rieche, and S. Gtz. Distributed Hash Tables. In *Springer*, 2005.

[27] C. Yang, W. Tarng, K. Hsieh, and M. Chen". A Security Mechanism for Clustered Wireless Sensor Networks Based on Elliptic Curve Cryptography. `http://www.my-smc.org/news/back/2010_12/main_article3.html`. Accessed June 13, 2013.

[28] R. Zilpelwar, R. Bedi, and V. Wadhai. An Overview of Privacy and Security in SNS. In *International Journal of P2P Network Trends and Technology(IJP2PNTT)*, 2012.

Streaming and live streaming in small and large networks

Christy Jovita Bibiana Arockiaraj
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email: achristy@mail.uni-paderborn.de

Abstract

Nowadays media streaming in internet applications has became very popular among the internet users in the past decade and it owes a lot to the Peer-to-Peer(P2P) technology for its tremendous growth. P2P came as a boon to internet users who can watch TV and live events via internet the minute it is telecasted. The streaming can be classified in two significant types (i) Video-on-Demand(VoD) and (ii) Live streaming. The essential element of a good streaming is the low startup latency, playback time video without any interruptions and quality of video. In this paper, we discuss about the approaches,challenges and security issues faced by streaming and live streaming applications in the internet.

1 Introduction

In the past few decades the world wide web is used by large number of users and it is increasing rapidly day by day. According to the survey conducted by [2],it was found that there were only 361 million users in the year of 2000 and it has increased to 1,967 million users in 2010. Due to increase in number of users,the network traffic and the client request for a single resource in the internet has also increased. Large scale multimedia streaming like watching live events and TV via internet the minute it is telecasted was a dream even a decade back. In the past few years streaming has seen a tremendous growth in the internet world. Streaming over the internet requires huge amount of resources and server utilization. Cisco predicts that by the year 2015,there will be more than 15 billion devices connected to internet and total amount of global internet traffic will be quadrupled by the year 2015[16].

Most of the traditional streaming techniques uses client-server based model.In client-server model,the central server handles all requests and it is very expensive since it needs a dedicated server.However,for small scale streaming media,few schemes like caching and replication techniques reduces the workload for the web servers and makes it scalable and fault tolerant[25]. For large scale streaming,traditional Content delivery network(CDN)[23] is widely adopted to overcome limitations of the internet in terms of quality,network performance,maximizing bandwidth and improving accessibility. A CDN replicates the content from origin servers to the cache servers and when client requests are sent , it is directed to the nearest cache server and that server delivers the content to the end user. The main drawback with CDN approach is its high cost[29].

The internet resources are concentrated in small number of nodes and it has to provide access with satisfiable response time all round the clock so the complicated load balancing and fault tolerance algorithm has to be applied and the same holds true for network bandwidth,which leads to bottleneck problem[4].These two main problems made the researchers to come up with Peer-to-Peer idea by distributing load and network network bandwidth among all participating nodes in the network[4].

P2P networking has emerged as a new technology to overcome most of the client-server model failure.It is easy to deploy and it is cost effective.One of the most popular P2P file sharing application is BitTorent[1] and some of the media streaming applications are SOPCast[2],PPLive[3],Zattoo[4]etc.In P2P networking, the users acts as both client and servers known as *peers*.The peers may join and leave the network frequently and whenever they want. The users not only download the content but also upload the content which is used by other users in the network.The peers shares their resources such as disk space or network bandwidth with the neighbouring peers. There is no centralized server for the whole network.Due to the decentralization behaviour,the single point failure or bottlenecks in the system are avoided.P2P networking is self-organized,distributed,fault-tolerant and load-balancing [8].

P2P file sharing application like BitTorrent has been been widely used for downloading file over the internet.

[1] http://www.bittorrent.com/
[2] http://www.sopcast.com/
[3] http://www.pptv.com/
[4] http://zattoo.com/

Now P2P technology has been developed to provide streaming services like Video on Demand and Live streaming. While downloading the content, the file(.torrent file) is split into pieces and the peer downloads the missing pieces from other peers and also upload the pieces which it has already. By this way all the peers in the network can download the file and the order of pieces is not important [6].

Unlike BitTorent file sharing, the video streams received and played at the same time. Video streaming is a time sensitive content and it plays the video while downloading,so the order of pieces being received by the peers are very important. If the peers receives the piece after the play-out time then the piece is of no use at that time and it reduce the quality of the video.Usually there is an end to end delay between the sender and the receivers and also requires high bandwidth.As the peers can join and leave the network at any time, it is important to take care of organizing the peers in a structure in the network.

However,locating the peers and maintaining the content delivery path are the main challenges in P2P technology [18].The P2P streaming network overlay topology can be classified into tree based and mesh based overlay.In the next section,the paper discuss about the two P2P overlay topologies used in the Video on Demand and in Live streaming services.In the section 4, the paper focus on the security issues faced by P2P streaming services.In section 5, we discuss about how social networking in P2P helps in enhancing the streaming services.The final section is the conclusion about P2P streaming.

2 P2P Video On Demand

Video on demand allows users to play,pause and stop videos whenever they want,which brings flexibility and convenience to the users. In VoD streaming, users from various parts of the world watch same videos and each user may watch the same video at different playback positions in the video.Unlike Live streaming,in P2P VoD the users can be able to do VCR operations like pause,forward or rewind the video.The users frequently leave the current playback position and jump to the new position.These frequent VCR-like operations cause long latency and high network overhead [32].This is the challenging design situation in P2P VoD systems.

Peers(Users) may stop watching the video and leave the network,if the Quality of video is reduced.Therefore,if quality of video is degraded there is more likely that peers can leave the system. This problems leads to the importance of robust failure recovery protocol in which the protocol used to reconnect the abandoned peers so that there is no loss of frames and delay in playback time[10].The system must also allows the peer to join the network at any instance of time and must provide the full-length video to the peers in the system.

The construction of peers i a topology is very important in streaming services,as it has to distribute the content between them.In the following section, the paper discuss about the tree based and mesh based overlays used in Video on Demand service.

2.1 Tree based systems

In tree based approach the peers are organized into tree structures where the root node serves as a server.The nodes in the tree download the stream only from its parent node and forward the received video to its child node.Traditional VoD service uses unicast model for streaming video [25].In this model every client setup its own connection to the server in a single channel. Based on the video popularity among the users,the server becomes the bottleneck and also consumes a lot of bandwidth.Many approaches have been used to solve the problems of scalability and network bandwidth.

The major challenge for P2P tree based system is to design an appropriate protocol to accommodate asynchronous users in the tree.P2Cast is one of the best example of tree based Peer to Peer VoD streaming systems.

P2Cast [12] uses a patching technique for streaming. P2Cast is extension of the IP Multicast patching technique.In P2Cast, peers arriving around the same time forms a session. For each session, server along with the peers(P2Cast clients) forms a application-level multicast(ALM) tree known as the *base tree* over the unicast only network. The server streams the entire video over the base tree so that it can be shared among all the peers. The peers not only receive the stream but also forward the stream to the other peers in the session.Peers who arrive later than the first one in the session misses the initial part of the video. The new peer joins the base tree and retrieves the stream from the base tree and also obtain the patch which is the initial part of the video(the part which was streamed from the start of the session to the time when the new peer joins the tree) from the server or from other earlier clients that have already cached the initial part of the video[12].Peers in the P2Cast systems provide two functions like

- **Base stream forwarding** : Forwarding the base stream among the peers and forms the Application level multicast tree
- **Patch serving** : Peers need enough storage capacity to store the patch and to serve it to other clients.

The Fig 1 illustrates the tree based(P2Cast) system.In practice,P2Cast follows 4 process for VoD service. The first process is the new client first contacting the server.The P2Cast maintains a Threshold T for a particular session.If the new client arrives within the threshold T,then it joins the session. Thus,the server and the clients in same session forms

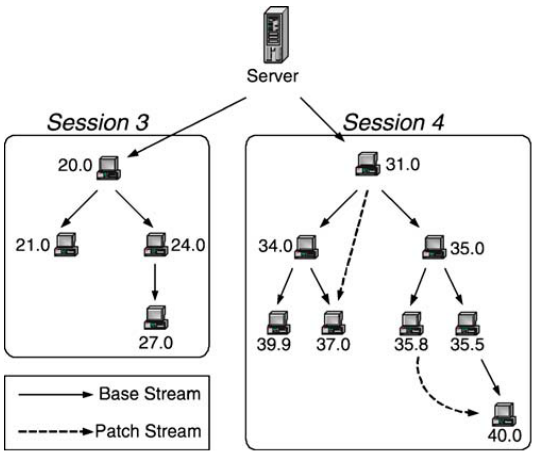


Figure 1. A snapshot of the scheme at time 40. Users belonging to the same session form an application-level multicast tree together with the server. Users in session 3 have finished patch retrieval; while 2 clients in session 4 are still receiving the patch stream from their parent patch servers[19]

a base tree.During the joining process,it has to select the patch server in its session from which it can retrieve the patch.For base tree construction and patch server selection it uses Best Fit Algorithm[12].

The second process is the base tree construction.When many P2Cast client arrives,it forms ALM tree and it uses tree-first approach [15] to build the tree. When the new client arrives,the base tree construction starts with the server.The server checks the bandwidth of the new client and finds whether it can be its child node.Otherwise it redirects the new client to its child node and this process goes on till the new client finds the appropriate parent node or it gets rejected.

The third process is the *patch server* selection.It is similar to base tree joining process,but it has to select the patch server until it finds.

The fourth process is the *failure recovery*.The client may leave the system,because of some network failure or computer crash.The departure of the clients may lead to disruption in the base stream transmission and the patching service also get distorted.This departure of client is detected by some kind of signal.

P2Cast provide two recovery mechanisms for base stream and patching. In base stream recovery if a client X is disrupted by a failure then only client X allows to contact the server and performs recovery by obtaining the base stream.If the recovery is failed,then the child of client X contacts the server for recovery.This process continues until it is recovered.In the patch recovery the client contacts the patch server that arrived earlier than that.Once the patch server is selected,the recovery is succeeded otherwise it has to contact the next patch server until it finds the appropriate patch server.

In the tree based system, the peers depend on its parent node for streaming content.Suppose if the parent node in the top level leaves the system after watching the video,then the lower part of the tree is left out.According to [27],the solution for the peer churns is the *Pre-fetching* technique in which the future content is also prefetched before the content is played out.

2.2 Mesh based systems

In mesh based P2P approach,the file is divided into small pieces of data.The server(seeder) distributes the data to different peers in the network.The peers downloads the missing piece of data from the neighbouring peers.BitTorrent is one of the best P2P file sharing protocol which uses swarming data transfer mechanism. To utilize the user's bandwidth to the fullest and to achieve high downloading throughput[19],it is better if different users have different pieces and there is always something to exchange.In case of video streaming the peers can be able to watch the video only if it receives the piece of data in sequential order and not in random order like in P2P file sharing.Due to the asynchronous requests from the users and the users interest in watching the video in different playback positions makes the designing of a mesh based P2P VoD scheme becomes more challenging.

BiTos[31] is one of the mesh based P2P approach for VoD based on BitTorrent protocol.In BiTos,only the piece selection mechanism is different than that of the BitTorrent,because in BiTos the arrival order of the piece should be sequential,which is very important for streaming.It uses three main components for streaming which is explained in 2:

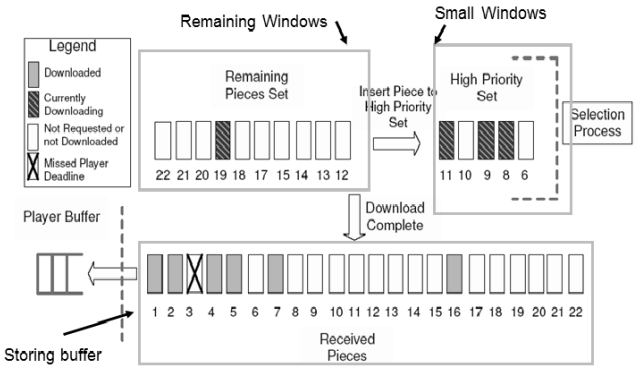


Figure 2. BiToS Approach [31]

- **Received Pieces**: Contains all the downloaded pieces

in order.The state of the piece may be downloaded,not downloaded or the piece that did not meet the deadline.

- **Higher priority set**: Contains the pieces that are close to be played in the player and they are currently in downloading or not-requested state.The pieces in this set has higher priority to be played next.

- **Remaining pieces set**:Contains the remaining pieces that are neither in higher priority nor in received pieces and they are in downloading or not-requested state.

The process involved in selecting pieces for any of the above three sets should be considered carefully.The peer selects a probability p to be downloaded which is in higher priority set and with the probability 1-p in the remaining set.The probability p can be adjusted dynamically for different conditions[31]. Rarest first mechanism is used to choose the piece from the higher priority set and the remaining set[6].

Large value of p indicates selection of more pieces in higher priority set that will be played soon and requested earlier than the pieces in the remaining set.In this case,peers choose to download that most of the peers have and would not have any rare pieces left to distribute and exchange among the other peers,which leads to choking. The probability p=0.8 using rarest first mechanisms works better,because it retrieves the rare pieces and also increase the diversity of the exchanged pieces[31].

In P2P VoD system[19],the newly arrived peers can receive the initial missed part of the video from the earlier arrived users if the video is downloaded by the earlier users in the order of playback. As time goes on , the peers watching the video gets decreased as they leave the network after finished watching the video.So it becomes a problem for the newly arrived peers because there is no one to share the video.

However,one of the approach called PONDER[13] intend to solve the performance issues faced by VoD service while using the VCR operations.In this approach it incorporates the mesh based P2P downloading into the conventional client server VoD service.A video is divided into equal segments and are exchanged among the peers like in BiTos.If the portion of the video segment is missing,then the missing part is served by the server.The server streams the data at the playback rate to ensure it is received on time.

3 P2P Live streaming

In live streaming the video content is broadcasted to all the requesting users in real time.Unlike VoD,the video playbacks are synchronized to provide same content to the peers.Large scale P2P live streaming systems have been successfully deployed in internet by providing different channels to millions of users across the world.Some of the popular P2P live streaming systems are SOPCast,PPLive,Zattoo.P2P live streaming uses two different overlay structures: Tree based and mesh based systems.

3.1 Tree based systems

In tree based P2P streaming applications,the peers forms a tree based structure and the child node receives the stream from the parent node like in P2P VoD tree based systems.Thus the video content is distributed among the peers.The P2P tree based delivery is originated from IP Multicast[3].IP Multicast reduces the traffic because it simultaneously delivers a single stream to thousand of clients,replicated by the routers along the multicast tree rooted at the source server.The use of bandwidth resources and the complexity of transport control for multicast sessions leads to overhead [19].Thus IP Multicast never became widely used in the internet.However this multicast function is implemented at application layer and is commonly known as Application-level-multicast.Unlike P2P VoD system tree based streaming,the users joining the tree start to watch the video content from the time they join the tree.There is no VCR-like operation.So there is no need for patching technique in tree based Live streaming.

3.1.1 Single tree Streaming

In single tree streamingYouyangchao scheme, similar to the IP Multicast the users participating in a video streaming session forms a tree at the application layer rooted at the video source server.Each user joins the tree at certain level and receives the video stream from its parent peer and also forward it to the child peer[19].End System Multicast[ESM] is the typical example of the Single tree P2P Live streaming.ESM research group at Carnegie Mellon pioneered the overlay multicast architecture and techniques to support scalable Internet video streaming[1]].

Figure 3 illustrates ALM tree.This ALM tree consists of ten peers.There are two peers in the level 1 which receives the video from the root node(Video server).These two peers forward the video to the next level which has four peers. Out of the four peers,only three peers forward the video to four other peers at the bottom level which is the leaf node.

While constructing a tree, the depth and the fan-out of the internal nodes are taken into account.The end to end delay from the video server to the peers in the bottom level may be large,because the content has to go through many intermediate peers in each level.Peers in the lowest level receives the video content at the last, so its better to have few levels possible while constructing the tree to reduce the delay in the bottom level.The tree structure should fan-out as much as possible.However,the peers in the internal node has constraints in uploading capacity at the full rate to its child peers[19].

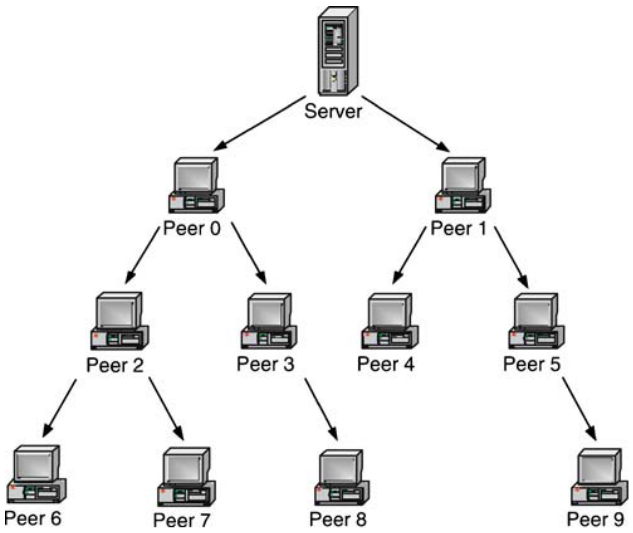


Figure 3. Application layer multicast tree for P2P video streaming[19]

Next important process is tree maintenance for tree based topology. Peers joining the tree in same session is dynamic and they may leave the system gracefully or unexpectedly. When a peer in the tree topology abandons the system, all the child nodes of that peer gets disconnected and cannot receive the video. The tree should be recovered from this problem as soon as possible to avoid the disruption in the video for the child peers.

Tree construction and maintenance[25] can be done in two possible ways : centralized or distributed.In centralized approach,the central video server controls the tree construction.When a failure occurs,the server detects the departure of a peer by sign-off signal or through some time-out signal.The server immediately recalculates the tree topology and forms the new tree topology. The main drawback of the centralized approach is the server may become bottleneck for large streaming system.

In distributed approach,ZigZag[30] algorithm is used for tree construction and maintenance.However,all the leaf nodes don't contribute their upload bandwidth,which degrades the peer bandwidth utilization efficiency [25] and the tree based streaming cannot recover to handle frequent peer churn.This drawback is overcame by Multi tree based streaming.

3.1.2 Multi tree based streaming

[19] In the multi tree based approach(refer Figure 3),the server divides the stream into many sub-streams and the participating peers are organized into multiple sub trees,one for each substream.Every peer joining the system joins all the sub-tree to retrieve the substreams.Every peer is placed in different levels or in different positions.It might be an internal node in one sub tree and a leaf node in another sub-tree.*Tree construction algorithm* is the most important part in multitree streaming systems. The tree constructed should be short trees.When a peer joins the system, it contacts the bootstrapping node and identifies the parent node.To maintain few levels in the tree,the new node should join the sub-tree which has few levels and is ready to accept the peer as one of the child or a leaf node[20]. Thus internal nodes are balanced among the different sub-tree.

When an internal node leaves the system,then the children of that node are partitioned from the base tree.The root of the base tree should rejoin with the partitioned node.If it is unable to join the tree, then all the nodes in the partitioned sub-tree independently start joining with the root node as an internal node or leaf[20].

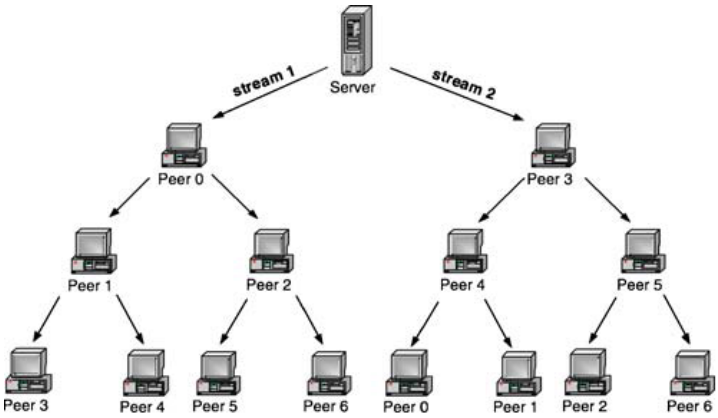


Figure 4. Multi-tree based streaming with two sub-streams and seven peers [19]

Unlike the single tree streaming,when the peer leaves the node the child node of that node still receive the sub-streams from other sub-tree.This makes all of its child nodes receive video streams in one way or other[25]. Usually the peers are an internal node in one tree and leaf node in s some other sub-tree which makes the upload bandwidth of an internal node can be utilized to upload substream over that tree.These are the main advantages of the multiple tree. Due to the frequent churns,the tree becomes saturated and cannot accept anymore new leaf nodes and leads to deadlock.When deadlock occurs,the leaf node tries to rejoin the tree until it succeeds[20].

3.2 Mesh based Overlays

In a mesh based streaming,it does not maintain a tree topology. The peers in the system does not have particular structure,rather it randomly forms a overlay or

mesh. The original video content is distributed among the peers.The peers exchange the data among each other by uploading or downloading.Many recent P2P live streaming applications like PPLive,sopcast adopts mesh based overlay.In mesh based system, the two main operations are taken into consideration:(i)Mesh formation and maintenance(ii)Data exchange.In *mesh formation and maintenance*,the tracker(bootstrapping node) keeps track of all the peers in a video session. When a peer joins the system it contacts the tracker and receive the peer list that will form the neighbour of the new peer.The peer list contains the random list of active peers in that video session. The new peer will try to make connection with the peers in the list and once the new peers obtains enough neighbours,it starts to exchange the data.Due to the frequent arrival and departures of peers, the mesh has to be reformed by updating the peer list.When a peer leaves the session gracefully or unexpectedly like computer crash,the peers notify the tracker by regularly exchanging the keep-alive messages.So that the peer will remove its information from its peer list.The second operation is the *data exchange*,in which the video is divided into small chunks,that are distributed among the peers.Every chunk is assigned a sequence number. The chunk with the lowest sequence number is the part of the video that has to be played at the earliest.The chunks are exchanged between the peers and it reaches the destination peer in non-sequential order.The video chunks are buffered into the memory where it is sequentially arranged,before the video content is being played in the media player.

There are three different data delivery techniques: mesh push, mesh pull, hybrid push pull.In mesh push systems,the peers push the content to its neighbouring peers.The main drawback of this system is, a peer might push the content to other peer which might already have the same video chunk or two different peers push the same video chunk to the same peer[19].The peer bandwidth may be wasted due to the redundant push.Mesh push schedules should be carefully planned.Anysee[5] is the typical example of mesh push based P2P live video streaming.

The second data delivery technique is mesh pull.In mesh pull system,the peers requests the video chunks from its neighbours.Thus,it avoids the redundant push.Each peer has a buffer map which contains the sequence numbers of chunks currently available in its buffer.The buffer map is exchanged among the peers periodically. The peer may then decide which video chunk it has to request and which peer it has to contact,after receiving the buffer map. But this frequent buffer map exchanges and the requests made by peers may cause signal overhead and additional delays in retrieving the video chunks.PPLive,SopCast and Coolstreaming[6] are the examples of mesh pull based streaming.The third approach is the hybrid mesh pull-push.It is the combination of push and pull technique,which has a high bandwidth utilization of pulling and low delay of pushing [5].

[5]http://www.anysee.com/

[6]http://www.coolstreaming.us/

4 Security and privacy issues

As the P2P streaming applications becomes more popular worldwide,it is exposed to more security and privacy threats.Compared to other P2P applications,the streaming applications are more vulnerable to the security attacks due to the QoS fluctuations[11].

P2P streaming applications depends on the peers in the network for its overall quality.If the peer does not receive the video content on time and if the quality is bad,they start leaving the system.This results in degradation of overall quality of the P2P streaming. The most serious attacks are caused by one or many of the peers inside the system network.When a peer joins the system,it is considered to be genuine and it gets all the privileges from the system.But in some cases,the peers intend to attack the system and pretends to be honest.This activity is called collusion.The peers causing the collusion is known as *colluder*,which is very hard to find in the system.

Some of the common attacks in P2P streaming approach are as follows:

- **Forgery attacks**[14]: The attacks that involve forgery and tampering of the data,streamed in live or VoD streaming service. To avoid this attack,message signature and public key infrastructures can be used. But in case of streaming the cost of signatures are expensive.

- **Pollution attacks**[9]:This attacks involves in distributing the polluted chunks in the video streaming.The attacker joins the system and establish connection with other peers in the system.When the attacker starts to exchange the streamed data,it exchange the polluted chunk to the neighbouring peers instead of the proper chunk.Unknowingly the neighbouring peer receives the polluted chunk from the attacker and starts sharing it with the remaining peers and contaminate the whole P2P system. The pollution attack may be caused by amateur attackers to degrade the quality of the channel for fun.Sometimes,if two channels are competing with each other, one of the channel may try to pollute the other.Chunk signing is the best defense mechanism used for pollution attack in Live streaming systems.

- **Sybil attacks**[11]: A single malicious node in a peer to peer network can disguise into multiple identities and gains trust from the neighbouring entities and start to control over some part of the network.Since it has the control over part of the network,it can pollute the network in many possible ways.Some of the solutions to Sybil attacks are discussed in [17].

- **Membership and eclipse attacks**[28]: The continuation of Sybil attacks is the Eclipse attack.Once the attacker gains control over part of the network,it grows until it has full control over the whole overlay.Thus the attacker gains the control of the overlay maintenance algorithm and the routing mechanism.The defending mechanism used for sybil attack is not enough for eclipse attack.To prevent the sybil attack,bounding the degree of overlay node is important.Usually the indegree and outdegree of the attacker node is above the threshold.So the legitimate peer should choose the neighbour from the subset of overlay node whose indegree and outdegree are below threshold.

- **Neighbour selection attacks**[11]:The peers establish a connection with the neighbouring peers by some selection mechanism.In this attack,the malicious node control the neighbour selection protocol mechanism.The attacker controls the traffic of the whole network by attacking the overlay communication and the selection mechanism.To prevent this attack,the invariants are identified in the node placement in the overlay and use deviations from invariants to detect the attackers.This approach is used in structured overlay like Distributed Hash table. The prevention of this attack in mesh based system is described in [26].

- **Dos attacks**[11]: This attack may cause different issues like sending excessive amounts of requests in short period of time or unwanted duplicate packets.Due to this DoS attack,some malicious node request more bandwidth and intentionally use all of the available upload bandwidth.This attack is also known as request spreading attack.To prevent this attack,some of the trusted nodes should store the download information of the remaining nodes and limit the nodes that consumes large amount of bandwidth[7].

- **Omission attacks**[11]:In this attack, all of the packet or part of it is not sent further according to the protocol specification. This type of behaviour compromise the whole system even if a small number of peers collude.

The main aspects in P2P streaming are overlay topology,data distribution and the peers in the system. The attacks most commonly focus on these three aspects as their targets. For example,the forgery and pollution attacks mostly concentrate on attacking the data distribution,while eclipse and sybil attacks focus on the overlay topology,Dos and Omission attacks on the peers[11].

In P2P streaming the overlay topology changes dynamically and reconstruction of the overlay happens often.So the importance should be given to the mechanisms involved in construction of the overlay and to protect them against any security attacks.The second aspect is the data distribution,which is concerned with how the data is exchanged and how the neighbours are selected by the peers.

In tree based approach[11],each node receives data from the root node which causes bandwidth fluctuations and the nodes closer to the root node turns into bottleneck. In tree based approach identifying malicious node is very difficult because if a correct peer receives a polluted chunk,it cannot decide if its parent node is malicious. The most common attacks on tree based approach is omission and forgery attacks. *Random monitoring* and *signed acknowledgement* are the solutions for these attacks. By introducing a single session trust manager which monitors all the peers,the peers in the system reports any complaints regarding the malicious nodes to the trust manager.If the trust manager cannot detect the location of the malicious node then ,it will decrease the trust on both the reported node and the reporter node.But this solution does not handle sybil attacks. The most common solution for tree based approach is given by Securestream[14].

In mesh based approach[11],there is no topological structure and no fixed flow(from root node to leaf node) like in tree based.The solutions to identify malicious nodes and to prevent the flooding and membership attacks are monitoring and audit schemes.Two examples of the mesh based overlay solutions are Prime and Coolstreaming.Overall, in peer to peer video streaming,the collusion based attacks are the most concerning issues.

5 P2P Streaming using Social Networking

As far,we have seen the approaches used in the P2P VoD and Live streaming.The overall system performance in P2P network highly depends on the peers.There are few issues in the previous P2P systems like the arrival and the departure of the peers are highly dynamic.The users leave the system as soon as they find the content not interesting.The next issue is the free riding.Some of the peers do not contribute to the upload capacity.Those peers are called Free riders[21] which degrades the overall performance of P2P system. Moreover,the peers joining the network is anonymous and they do not have any social relationships between them.Unless the peers are known to each other,it is very difficult to trust the peers in the network.Because of these issues,it is a better idea to combine social networking with P2P system. So that the peers interact with each other,make contact and forms a trusted network group.

Tribler[24] is one of the application using social based P2P networking.In this Tribler application, it overcomes few challenges like decentralization,availability,integrity,trust and network transparency which were some of the few challenges faced by BitTorrent.Tribler is built based on BitTorrent with additional social networking feature and it is also fully decentralized.

One more advantage of social based P2P networking is the willingness to share their resources towards their friends in their group.This makes the count of free riders less.By building the P2P overlay above the social network is safe and secure for content distribution.If the peer is identified as an attacker,it is removed from the system and also from the friends list.The peers are easily evaluated based on the past activities on the P2P system.The peers in a social based network cannot forge their identity and disguise into some other identity as it uses permanent Id. By taking the advantages of these social networking features,and using an incentive mechanisms such as Tit-for-tat mechanism[6] or Give-to-get[22] mechanism,it is able to stream VoD or Live streaming in P2P systems securely.However, one of the main issue that has to be addressed is the end to end delay and low latency.

6 Conclusion

In this paper,we discussed about the peer to peer streaming technology,its security issues in streaming and how it can be implemented in social networking.Live streaming and Video on demand services over the internet has gained more popularity in the recent years.The peer to peer streaming applications provides service to large number of populations at low cost and with simple infrastructure.However, the quality of the video streamed is not as good as in traditional TV services.In P2P streaming,the video streaming depends on the number of users watching the video.If the number of viewers is very small,then the video quality is poor and unstable.So the unpopular content are not suitable to view in Peer-t-Peer streaming.This is one of the challenging issue in peer to peer streaming systems.However, the Peer-to-Peer streaming is one of the promising approach in the world of video streaming.

References

[1] Conviva. http://www.conviva.com/the-science/.

[2] Internet World Statistics. http://www.internetworldstats.com/.

[3] IP Multicast Technology Overview. http://www.cisco.com/en/US/hmpgs/index.html.

[4] K. Aberer and M. Hauswirth. Peer-to-peer Information Systems: Concepts and Models, State-of-the-art, and Future Systems. In *ACM SIGSOFT '01:Proceedings of the European Software Engineering Conference on Foundations of Software Engineering*, 2001.

[5] K.-H. K. Chan, S.-H. G. Chan, and A. C. Begen. Optimizing Substream Scheduling for Peer-to-Peer Live Streaming. In *Proceedings of the IEEE conference on Consumer communications and networking conference*, 2010.

[6] B. Cohen. Incentives Build Robustness in Bittorrent. 2003.

[7] W. Conner, K. Nahrstedt, and I. Gupta. Preventing DoS Attacks in Peer-to-Peer Media Streaming Systems. In *Autonomic and Trusted Computing*, 2006.

[8] N. Daswani, H. Garcia-molina, and B. Yang. Open Problems in Data-Sharing Peer-to-Peer Systems. In *Proceedings on the International Conference on Database Theory*, 2003.

[9] P. Dhungel, X. Hei, K. W. Ross, and N. Saxena. The Pollution Attack in P2P Live Video Streaming Systems: Measurement Results and Defenses. In *ACM : Proceedings of the workshop on Peer-to-Peer streaming and IP-TV*, 2007.

[10] T. T. Do, K. A. Hua, and M. A. Tantaoui. Robust Video-on-Demand Streaming in Peer-to-Peer Environments. *Computer Communications*, 2008.

[11] G. Gheorghe, R. Lo Cigno, and A. Montresor. Security and Privacy Issues in P2P Streaming Systems: A Survey. *Peer-to-Peer Networking and Applications*, 2011.

[12] Y. Guo, K. Suh, J. Kurose, and D. Towsley. P2Cast: Peer-to-Peer Patching Scheme for VoD Service. In *Proceedings of the WWW Conference about Multimedia Tools and Applications*, 2003.

[13] Y. Guo, S. Yu, H. Liu, S. Mathur, and K. Ramaswamy. Supporting VCR Operation in a Mesh-Based P2P VoD System. In *IEEE CCNC '08:Proceedings of the Consumer Communications and Networking Conference*, 2008.

[14] M. Haridasan and R. V. Renesse. Defense Against Intrusion in a Live Streaming Multicast System. In *IEEE '06:Proceedings of the International Conference on Peer-to-Peer Computing*, 2006.

[15] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas. A Survey of Application-Layer Multicast Protocols. In *IEEE CST '07 : Proceedings of the Communication Surveys and Tutorials*, 2007.

[16] S. Jose. http://newsroom.cisco.com/.

[17] B. N. Levine, C. Shields, and N. B. Margolin. A Survey of Solutions to the Sybil Attack. Tech report, 2006.

[18] B. Li and H. Yin. Peer-to-Peer Live Video Streaming on the Internet: Issues, Existing Approaches, and Challenges. *Communications Magazine, IEEE*, 2007.

[19] Y. Liu, Y. Guo, and C. Liang. A Survey on Peer-to-Peer Video Streaming Systems. *Peer-to-Peer Networking and Applications*, 2008.

[20] N. Magharei, R. Rejaie, and Y. Guo. Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches. In *IEEE INFOCOM 2007 : Proceedings of the International Conference on Computer Communications*, 2007.

[21] J. J. D. Mol. *Free-riding Resilient Video Streaming in Peer-to-peer Networks*. 2010.

[22] J. J. D. Mol, J. A. Pouwelse, M. Meulpolder, D. H. J. Epema, and H. J. Sips. Give-to-Get: Free-riding-resilient Video-on-Demand in P2P Systems, 2008.

[23] A. K. Pathan and R. Buyya. A Taxonomy and Survey of Content Delivery Networks. 2006.

[24] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. V. Steen, and H. J. Sips. Tribler: A Social-based Peer-to-Peer System. In *IPTPS06 : Proceedings of the International Workshop on Peer-to-Peer Systems*, 2006.

[25] T. Sabu. A Review on P2p Video Streaming. 2013.

[26] J. Seibert, D. Zage, and C. Nita-Rotaru. "Won't You Be My Neighbour?" : Neighbour Selection Attacks in Mesh-based Peer-to-Peer Streaming. 2008.

[27] A. Sharma, A. Bestavros, and I. Matta. dPAM: A Distributed Prefetching Protocol for Scalable Asynchronous Multicast in P2P Systems. In *IEEE INFOCOM '04:Proceedings of the Annual Joint Conference on Computer and Communications Societies*, 2004.

[28] A. Singh, M. Castro, P. Druschel, and A. Rowstron. Defending Against Eclipse Attacks on Overlay Networks. In *ACM SIGOPS :Proceedings of the European workshop*, 2004.

[29] K. A. Skevik, V. Goebel, and T. Plagemann. Analysis of Bittorrent and its Use for the Design of a P2P based Streaming Protocol for a Hybrid CDN. 2006.

[30] D. A. Tran, , K. A. Hua, and T. Do. ZIGZAG: an Efficient Peer-to-Peer Scheme for Media Streaming. In *IEEE INFOCOM 2003: Proceedings of the Annual Joint Conference of the Computer and Communications Societies*, 2003.

[31] A. Vlavianos, M. Iliofotou, and M. Faloutsos. BiTos: Enhancing Bittorrent for Supporting Streaming Applications. In *IEEE INFOCOM '06: Proceedings of the International Conference on Computer Communications*, 2006.

[32] T. Xu, J. Chen, W. Li, S. Lu, Y. Guo, and M. Hamdi. Supporting VCR-like Operations in Derivative Tree-Based P2P Streaming Systems. In *IEEE ICC09 : Proceedings of the IEEE International Conference on Communications*, 2009.

Sharing large files through Bit Torrent in Peer-to-Peer social networks

Kailaash Balachandran
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email: kailaash@mail.uni-paderborn.de

Abstract

Bit-Torrent (BT) is a widely used protocol for file sharing over the network in an efficient way where a single file is effectively accessed by large number of users. Bit-Torrent is based on peer to peer networking (P2P) and it improves the transfer speed drastically by splitting the file into segments and collecting the segment needed from the other peers that already have them. The Bit-Torrent technique shows effective resource utilization among hundreds of connected peers accessing the same file by not compensating on their download speed. This paper discusses the working principle of Bit-Torrent , framework behind its operation, proposed modifications in its working and what the future might hold for this innovative approach in order to serve files over the peer to peer social network.

1 Introduction

In recent times, Bit-Torrent has become one of the most popular peer to peer content distribution technique over the web and it has higher share of Internet traffic than all other peer to peer networks. Bit-Torrent was installed on 28.20 percent of computers worldwide in September 2007 and this number is certainly higher given the current trend of Torrent usage [1]. A recent Internet traffic trend report released by the Canadian broadband management company Sandvine reveals that bandwidth usage patterns during peak hours of Internet usage show that of 29.97 percent of the upstream stream can be attributed to Bit-torrent [2] and the average download speed is about 30 KB/s, which is quite a acceptable rate to get large files in a day [3]. Bit-torrent's popularity has paved way for a number of third party torrent clients available for free download which provides ease to use interfaces and one click download feature to their users, which, on the other hand, contributed to the success of Bit-Torrent.

The Bit-Torrent's operation is based on hybrid decentralization of the content which makes it different from the traditional server client download technique. In traditional file sharing technique, to download a file, the client ,usually ,the browser requests the server that holds the file to transfer a copy of the file to the clients system. This transfer is handled by protocols such as FTP (File Transfer Protocol) or HTTP (Hyper Text Transfer Protocol). Here, the download speed is greatly affected by a number of factors such as amount of traffic on the server , the protocol used for the transfer and also the number of computers trying to download the file at the same time. The transfer speed varies depending on the demands of the server. However , in peer to peer file sharing method, a P2P client is used on every participating nodes. The client sends out a request for the file to download and to locate the file, the client queries other nodes that are connected. When the client finds a node that have the requested file,the download begins. The shared file is chopped into pieces and every computer that downloads these pieces, also uploads them to each other. These pieces are replicated on different computers as soon as possible. Once a computer has a complete piece, it is then traded with other computers that needs them. Thus every computer that participates in the transfer acts as both, server and client. A major factor behind Bit-Torrent's success is the built-in incentives mechanism, implemented by its choking algorithm, also called the Tit-for-tat algorithm which encourages every active node to upload pieces to each other. The file transfer load is shared among the computers exchanging files, but can cause bottlenecks when a node downloads the file and immediately disconnect without allowing others to obtain files. This limits the number of computers the software can search for the requested file. Here, we'll discuss the working principle of Bit-torrent and how it solves these problems.

The rest of this paper is organized as follows. Section 2 provides a description of the terminologies, operation of the Bit-Torrent protocol and an explanation of its piece selection and peer selection policies as implemented in the official Bit-Torrent client. Section 3 describes the drawbacks with conventional method along with the proposed and de-

ployed modifications to bit-torrent's framework. Section 4 provides an overview on file sharing in a collaborative P2P social networks, introduces the new paradigm of friend to friend network and lists out challenges faced in implementing an effectve file sharing feature. Section 5 presents bit-torrent in P2P social network by providing a basic paradigm on how a bit-torrent system reacts in a social environment and lists some of the social community biased optimizations to its framework that can be adapted in our project. Section 6 concludes.

2 Bit-Torrent Operation

2.1 Terminologies

Here, we define the terminologies used throughout this paper though they are not of standardized definitions.

- Peer: A peer is a active computer node that participates in the torrent file transfer. A peer can be in one of the two states. The leecher state and the seed state.
- Seeder: A seeder is a peer that has a complete copy of the torrent and still offers it for upload. The more the number of seeders, the better are the chances of getting a higher download rate.
- Swarm: The group of peers active on a torrent. Active means those peers that transfer content associated with the torrent either as a seed or a peer.
- Leecher: Leecher is a peer that still downloads pieces of the content and becomes a seed once it gets the complete file.
- Tracker: A tracker is a server that keeps track of which seeds and peers that are in the swarm. The tracker is centralized and a new peer wishing to join the torrent, contacts the tracker to get the list of peers that have the pieces of the file.
- .Torrent file: The .torrent file, also referred to as a Metainfo file, has all informations necessary to download the content and the number of pieces. A torrent file does not contain the content but only the information about those files, their names, sizes, structure, and the hash values for verifying file integrity.
- Index: An index is a list of .torrent files published and managed by a website. It acts as a search engine for users to search and download .torrent files.

2.2 Torrent Client

A Torrent client is a software program that facilitates peer to peer based file sharing over Bit-Torrent protocol. Most of these Bit-torrent clients are free and open source softwares. They provide a clean, user friendly interfaces

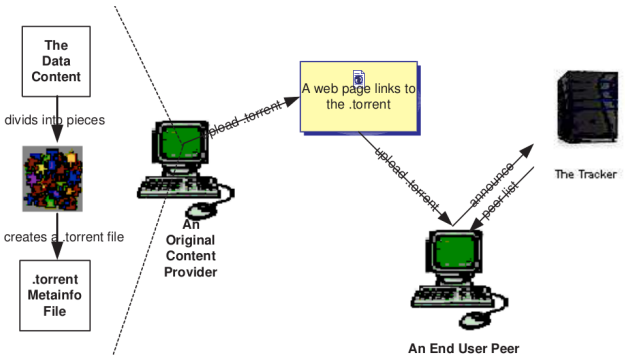


Figure 1. Overview of Bit-Torrent's Operation, Source:[4]

where the users launch by clicking on a hyper-link and are given a standard 'Save as' dialog [5]. These torrent clients vary in their functions providing unique features and also in their design framework which shows difference in its operating behaviors. The paper [6] , observes the performances of two major famous clients, namely uTorrent and Vuze, where uTorrent users achieve more download speed than the users of Vuze for the same set of torrents [6]. Performance of each client differs mainly on (a) How they manage their neighborhood size (b) How to find new connected peers (c) When do they close a connection and (d) How do they distribute upload capacity. Thus, the design choices and framework of bit-torrent clients have significant effect on the download speed.

2.3 Content Publishing

When a user decides to share a file using bit-torrent, the peer distributing the file uses torrent client to chop the file into identical sized pieces, creates hash for every single piece and is recorded in the .torrent file. Peers that has the complete file are called seeders, and the peer providing the first copy, generally the torrent owner, is called the initial seeder. These Torrents have an "announce" section where URL of the tracker is mentioned followed by an info section containing name of the files, number of pieces, their size and hashes for each piece. These are verified by clients to check integrity of the data that they receive. Torrent files are typically uploaded on websites such as index sites, with at least one tracker. The tracker maintains lists of the peers currently participating in the torrent. The initial seeder who has the complete file should start seeding to make the file available for other connected peers to download.

2.4 Role of a Tracker

A tracker is a server that helps in the communication among peers participating in the transfer [7][8]. To make it simple, a tracker just keeps track of what is happening on

the network. Peers regularly update the tracker to negotiate with the new peers and they contain the complete statistics of the transfer. When users create a torrent file, they would need to specify a tracker URL to the announce section in the client. The more trackers you specify, the faster the download speed will be. Some index sites provide their own trackers where it also hosts the torrent file providing all needed features to start a torrent download. Torrent clients also allow its users to create their own tracker by fetching the IP address of the system and the listening port number followed by announce section.

2.4.1 Public Tracker

The public tracker, as the name specifies , is open to public, Any one can use the tracker where the DHT and local peer discovery are enabled in order to spread far and wide. Public trackers are more prone to Free riding, where peers benefits from resources (download) without paying for the cost of the benefit (upload). In short, free riders are those selfish peers that only download pieces without contributing to its reciprocal upload.

2.4.2 Private Tracker

The defining factor of private bit-torrent trackers is that they're selective about who gets to use the site. Unlike public trackers, on which anyone can search and download torrents, private trackers require you to be invited by another user or create an account in order to use the service. Each site using the private tracker will have their own set of rules about who and how the site can be accessed. Because of these rules, you get a lot of benefits, including fast download speed, quality content, privacy and a great community.

2.4.3 Multi-trackers

Multi-tracker torrents contain multiple trackers in a single torrent file. This provides redundancy in the case that one tracker fails, the other trackers can continue to maintain the swarm for the torrent. One disadvantage to this is that it becomes possible to have multiple unconnected swarms for a single torrent where some users can connect to one specific tracker while being unable to connect to another. This can create a disjoint set which can impede the efficiency of a torrent to transfer the files it describes [8].

2.4.4 Tracker-less torrent

Even though bit-torrent works on decentralized P2P network, it highly relies on the centralized tracker servers. The simple solution is to go tracker less [9], which means there is no central tracker server. DeCentralized or Trackerless torrent is discussed elaborately in section 5.2.

2.5 Piece Selection Stratergy

In torrents, the file to be shared is broken into pieces and further into sub pieces and these pieces are negotiated and transferred between the peers. When a peer starts to download, it defines free spaces for the whole file, then it searching for different pieces of the file from the connected peers to download. Peers download these pieces in a random order , not necessayily from start to end piece. Adopting a appropriate piece selection strategy plays a major role in the overall performance.

2.5.1 Strict Priority

A single piece may have a number of sub pieces associated with it and different peer may contain these sub pieces. When a particular peer requests for a single sub piece, all the remaining sub pieces of the parent piece are requested before requesting any other sub pieces of other pieces. This policy helps downloading a complete piece as quickly as possible.

2.5.2 Rarest First

After allocating required space for the whole file, the peer now searches for pieces to download from connected peers. While selecting the pieces, the peer usually selects the piece which is found in rare in its connected network. This rarest first technique makes sure that all pieces are available on the network and gets replicated such that the overall performance is not affected even when a seeder who has the complete file goes off line. In some practical deployments, the initial seeder goes offline due to cost measures leaving the current leechers to exchange and share pieces among each other. In such cases, rarest first approach plays a significant role by not loosing a piece to go out of the network.

2.5.3 Random piece first

This is an exception to rarest first approach. When a peer starts the download, it has got nothing to upload for the other peers. The rarest pieces are present generally on fewer peers and it takes longer download time as it rate will be slow. So the peer selects a random piece and downloads it as quickly as possible so that it gets its first complete piece , thus joining the swarm of leechers and starts to upload to other peers. Once it gets the first complete piece, it adopts to rarest first technique.

2.5.4 End Game Mode

When a peer is about to complete the transfer, it will left with fewer number of pieces to download. Sometimes those pieces will be requested from a peer with slow transfer rate rather than a peer with higher rate. It will drastically delay

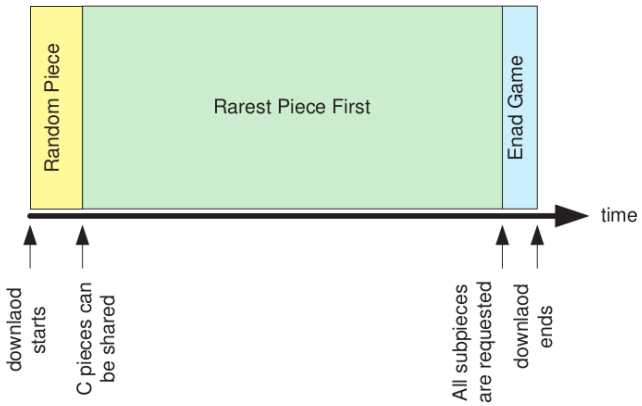


Figure 2. Combined overview of piece selection stratergies used by a peer from download start to end, Source: [4]

Figure 3.

the download finish. In end game piece selection algorithm , the peer sends out request to all peers for the sub pieces that is needed to complete the download. Once the piece gets downloaded from peer that has higher transfer rates, cancel messages are sent to all appending peers to avoid redundant sends. The end game mode is very short and doesn't have much impact on the overall download process [10].

2.6 Peer Selection Stratergy

Bit-torrent is a decentralized network and it does no central resource allocation. Each participating peer should be responsible to maximize their download rate [5]. Peers download pieces from different peers and upload simultaneously to a limited number of peers. In general economics, Pareto efficiency refers to the context where no two counter parties can make an exchange and both be happier at the same time. Seeking pareto efficiency is a local optimization algorithm in which pairs of counterparties see if they can improve their lot together,and such algorithms tend to lead to global optima. Specifically, if two peers are both getting poor reciprocation for some of the upload they are providing, they can often start uploading to each other instead and both get a better download rate than they had before. Choking algorithms in bit-torrent allows a peer to select other peer to which they can exchange data thus achieving Pareto efficiency [5].

2.6.1 The Choking Algorithm

In a bit-torrent environment, every peer need to download and upload at the same time. To cooperate peers upload, to not to cooperate they choke [5]. A choked peer doesn't upload any pieces and in other words, choking is analogous to temporary refusal to upload to other peers, but download continues to happen . Sometimes it happens when a peer with a good transfer rates chokes itself affecting the circulation of pieces in the network. This choked peer can download data faster due to its high transfer rates and donot upload any piece in return. As a consequence, free riders, i.e., peers that never upload, should be penalized, hence those peers with good transfer rates should be unchoked. When a peer is unchoked, it cannot choke itself. A peer has to decide about the peers that should be unchoked. Each bit-torrent peer unchokes a number of peers, usually four peers (by default) and the number is limited to avoid TCP congestions over the network. A peer decides about which peer to unchoke purely based on their transfer rate which is calculated for every 10 seconds. Those peer that show good transfer rates gets unchoked and re-unchoking happens when the next ten second period is up. Choking Algorithms differs for seeders and leechers. A seeder usually chokes all peers in a round robin fashion as they have the complete file and does not require any uploads to them.

2.6.2 Optimistic Choking

Regular Unchoking method may miss out some unused connections that have better transfer rates than the ones that are being unchoked. To fix this issue, bit-torrent uses optimistic unchoking mechanism where a peer is unchoked in random and the optimistic unchoke is recycled every third rechoke period , which is thirty seconds ,claiming that thirty seconds is a long enough period of time for TCP to ramp up new transfers to their full capacity and new connections are three times as likely to start at the current optimistic unchoke as any other connection in the rotation. The optimistic unchoke peer selection has two purposes. It allows to evaluate the download capacity of new peers in the peer set, and it allows to bootstrap new peers that do not have any piece to share by giving them their first piece [11] [12].

2.6.3 Anti Snubbing

Sometimes a peer with higher transfer rates will get unchoked for longer duration affecting its download speed. This happens when optimistic unchoke does not find any peer with better upload speed. Such peer , that receives nothing from a particular peer for more than one minute, is assumed to be snubbed and only a optimistic unchoke can release the peer from being snubbed . This results in more than one optimistic unchoke to recover the download rates of the snubbed peer.

3 Proposed Modifications in Bit-Torrent's working

Bit-torrent systems are highly popular and are considered to be a viable alternative to traditional client-server

model as they are decentralized. As the content is not stored in a central server, every participating peer is dependent on each other as they download and upload pieces of file . Even though Bit-torrent provides incentives to ensure fairness among participating peers, there exists peers called free riders who downloads resources from other peers but do not share (upload) to any. This causes degradation in the overall performance of the system. Also despite the implementation of the Tit-for-tat principle, the fairness among peers doesn't seem to be impressive. Much research has been done to improve fairness among peers but it has been observed that peers with high upload bandwidth frequently upload much more data than they download, with the opposite being the case for peers with low upload bandwidths [11]. The following algorithms have been proposed to improve the working of bit torrent.

3.1 Conditional Optimistic Unchoke

In Conditional Optimistic Unchoke, the Instantaneous Fairness Ratio (IFR) is calculated for for an individual peer as the ratio of data uploaded to data downloaded during the last 10 seconds. Therefore, an IFR of less than 1 indicates a peer is downloading an excessive amount and an IFR of greater than 1 indicates a peer is downloading an insufficient amount. Optimistic Unchoke happens only if the peer's IFR is greater than 1. The Peer need to be cautious for not downloading excessively and choking other peers may change the set of peers from where its downloading thus reducing its download amount [11].

3.2 Variable Number of Outgoing Connections

Instead of having fixed number of connections for each peer, this method proposes having variable number of connections depending on the current upload capacity of the peer. A peer with high upload capacity might have connection with more number of low capacity peers so that they can serve them better whereas a low capacity peer connect to limited number of peers. A pair of peers have multiple connections with each other thus enabling peers with more capacity can transfer data to one another at high rates[11]

3.3 Treat Before Trick

Treat Before Trick solves the problem of free riding to some extent. Each file is encrypted by the owner or the seeder. The key used in the encryption mechanism is again split into 'n' sub keys and pieces are sent along with sub keys, i.e each piece has a subkey. These sub keys have no dependence on the piece they have been sent. The le owner uploads both the le pieces and subkeys to a set of requesting peers, called leechers. Now the leechers can barter pieces for subkeys with each other such that every leecher will have to upload the pieces in order to get the subkey to decrypt the piece.The rule Data First Key Later is followed by peers thus making every peer to upload a piece before requesting for the key. Here, peers swap le pieces for subkeys and vice versa, which are needed in order to decrypt the le pieces [13].

3.4 Super Seeding

This technique helps when we need to seed a large torrent with limited upload capacity. Seed acts as normal leechers by not claiming to have the complete file. When the seed gets connected to leechers, it uploads a piece to a leecher that has been never uploaded before and does not upload another piece until it gets acknowledgement from other leechers that the sent piece has been uploaded again. This helps great for the seeds that have less upload capacity or have to pay for the upload bandwidth as this method limits the number of duplicated pieces[14]. Super seeding can help save an upload ratio of around 20

3.5 Tracker Protocol Extension

When a peer enters a torrent, it is called a leecher since it does not contain any piece. It contacts the tracker for the random list of peers and to achieve more optimal bandwidth utilization, it connects to the peers that have similar transfer rates. If there are very less peers with similar rates are present, it takes time to discover them and the only way to do so is by optimistic unchokes. It would be more preferable if the tracker that the peer initially connects to can perform this operations as it might save more time rather than waiting for an optimistic unchoke. The new peers that contacts the tracker to join the torrent, reports its upload capacity to the tracker and the tracker will reply with list of peers with similar transfer capacities. Optimistic Unchoke is then performed every 30 seconds so as to discover new partners sooner with same upload capabilities[14].

3.6 Team Enhanced Protocol

This is a slight modification over the tracker protocol extension. Trackers coordinates peers to form teams with same upload limit while joining the torrent. In addition to that, tracker enforces rules for a peer joining the team in order to avoid free riding. Firstly, it limits a peer to change teams frequently by tracking its history of team joins within a short interval of time. Secondly, the tracker do not allow a peer to join a team if it doesnt satisfy or found noncompliant with the team protocol in the past. For example if the peer had lied about its upload bandwidth before , the peer is blacklisted by the tracker and is not allowed to join in the team. Thus it prevents the entry of a peer that tries to free ride [4].

4 File Sharing in a social P2P environment

Online centralized file sharing social networks certainly have limitations over privacy concerns. Particularly privacy and protection from massive data-mining and big-brotherly

treatment of the users by the social networking service providers. By mapping a peer-to-peer application to a peer-to-peer infrastructure, direct connections can be exploited such that locality can be taken into account. Even though peer-to-peer file sharing systems do exist, they focus more on the technical aspects and therefore unable to exploit the power of social phenomena. Even though the bit-torrent methodology has been proven successful, it is still vulnerable to free riding and tracker offline scenarios. Integrating Bit-torrent technology in a peer-to-peer social network faces a lot of challenges where security plays the major role.

4.1 F2F Network

Friends-friends network is a new paradigm after peer-to-peer(P2P). In an F2F network [15] two computers can communicate with one another only if their owners are related in the social network. Since the two owners know one another they can communicate either in person, through an instant messaging service or any other means. This side channel can be used to set up the connections between users computers, i.e. by exchanging their network addresses and possibly some cryptographic data for securing the link.

4.2 Social trust becomes P2P trust

One of the biggest and largely unsolved problems of P2P systems is security. Because any peer can connect to any other peer this makes the systems extremely vulnerable. Peers can arbitrarily deviate from the P2P protocols, appear in the network under many identities or selfishly use the resources of others while not contributing their own. Even when peer misbehavior is detected the rogue peers need to be isolated from the network, which is extremely hard to do without having a stable peer identity. Open P2P systems are essentially groups of strangers talking with each other, trust is volatile and difficult to build. In F2F networks the peers have a well defined identity and the users bring the social trust into the system. A user in the system knows that the computers that her computer is communicating with belong to her friends and can be trusted to work as expected. If one of the computers does misbehave and this is detected, the users can be notified about it and undertake the necessary action, e.g. possibly contact the owner of the rogue computer. Strong identity allows for self-policing and user-mediated repair in F2F systems. F2F greatly reduces most of the P2P systems vulnerabilities by bringing the social trust into them. This happens at a cost though. The classical P2P systems rely on being able to connect to any peer in the network while in F2F, connections are only possible among friends.

4.3 Challenges faced in implementing file sharing feature in P2P social networks

- The Availability Problem:Guaranteeing the availability of P2P system is a major challenge in implementing file sharing feature. The system should not rely on the availability of seeds or any central resource manager. In [16], it is estimated that less than 4 percent of the peers have an uptime of over 10 hours. More number of peers tend to leave the system quite often leaving the system with less seeders or no seeders in worst cases. Hence, the availability problem is critical. Providing more incentives such as social recognition and awards could make users to run the P2P software for longer duration, thus improving the availability of the system as a whole.

- System Integrity: Maintaining the integrity of the system is the another major concern and how well the system can be trusted. The only solution for this is to achieve trust among peers. Free riders are the so called peers that do not cooperate with the system by selfishly not donating resources to other peers. So all peers cannot be trusted and maintaining system integrity has proven to be a great challenge in operational systems. This problem could be solved in a social based P2P network as users know each other or the group and can also select trusted users as representatives [17].

5 Bit-Torrent for P2P Social Networks

5.1 Private Trackers

Usage of private trackers, though centralized , provides greater benefits include larger selections of files, faster downloads, higher quality files, and strong communities behind them. They come with a lot of rules and regulations. These rules may include Rules regarding ratio which corresponds to upload/download ratio, rules regarding who can access the site, rules regarding the quality of the file and even more, which, in other hand improves wide selection of nonpoisoned torrents and provides a lot of benefits such as

- High Quality Files: Because of the rules, only high quality torrents gets uploaded. If a user uploaded torrent doesn't satisfy the quality rule, it is deleted automatically.

- Faster Downloads: The rule of Private torrent is all about seeding and have mechanisms to prevent free riding resulting in faster downloads..

- Increased Privacy: Here, torrents are kept private from the public eye. Compared to other public torrents, private torrent sites are less traveled and users are forced to register or get invite to access the site, which again improves privacy.

5.2 DeCentralized Trackers

Bit-Torrent is a peer-to-peer approach and even though it is claimed to be decentralized, the download process still

relies in part on central servers called trackers that can crash or go offline for a variety of reasons. To address this vulnerability, bit-torrent downloads should no longer require a central server, meaning that no central trackers. In todays deployment, Tribler client allow users to search for torrents and download files without the need for any central server, but the current client still lacks a solution for decentralized tracking of swarms . In the paper [9] , they have proposed completely removing the tracker and replacing it with a set of distributed protocols based on random walks, accomplishing the work of the tracker without explicitly tracking every peer in the system.

The distributed tracking algorithm , 2-Hop TorrentSmell, is used in the implementation of Tribler Client [18] . Modern bit-torrent clients can exchange their neighborhood sets through Peer Exchange (PEX). PEX greatly reduces the reliance of peers on a tracker by allowing each peer to directly update others in the swarm as to which peers are currently in the swarm. In 2-Hop TorrentSmell algorithm [19] , we track peers in the swarm by RePex (Reconnect to peers encountered before) and find the distributed trackers tracking the swarm. Much research has been done in the field of decentralizing trackers but they are not deployed anywhere officially outside the research domain. If such system comes in the future, there is definitely no way for anyone to monitor, ban or bring down a P2P network as it is completely decentralized.

5.3 De-Indexing

There are bit-torrent search engines and indexes such as The Pirate Bay and isoHunt. They search for the requested content and download the .torrent file. Decentralization of the index search can be done using the gossip protocol [20] that can act over the bit-torrent engine. The searched key word will be passed on to the neighboring connected nodes like a gossip. It spreads on throughout the swarm returning the result of the search. Gossip protocols are designed to operate in very large, decentralised networks. A peer to peer network keeps changing rapidly Ias new nodes arrive and existing nodes depart after download or even abruptly. In such cases, , gossip protocols perform well as information exchange happens in a random periodic way. Loss of data wouldn't affect the system as no nodes are assigned with specific tasks and are not given any roles . In spite of loosing a data, copies of the same can be received from multiple nodes. If a node fails or goes off line, it doesn't affect other nodes from sending messaged thus there won't be any single point of failure in the system.

5.4 BuddyCast algorithm

The list of content a person downloads via bit-torrent can be considered the taste of the user.Through its downloads the user builds up a preference list of content. The preference list contains by default all downloaded les from which the user can add or remove entries. These preference lists are exchanged freely amongst peers using the Buddycast algorithm. Using this algorithm the user builds Cache. The recommender component uses the Preference Cache to calculate both similarity between peers and to recommend certain content the user is predicted to like, using a special collaborative ltering algorithm. When a certain peer has a preference list with high similarity to the users they have the same download taste. We call such similar peers taste buddies. [17]

To enable effective content based discovery, every peer provides a list of preference of files , by default, with recently downloaded files. In Buddycast algorithm, Each peer maintains a list of its top-N most similar peers along with their current preference lists. Periodically, a peer connects to either (a) one of its buddies to exchange preference lists (exploitation), or (b) to a new peer, randomly chosen, to exchange this information (exploration). To maximize the exploration of the social network, every peer also maintains a list with the K most recently visited random peers, and avoids reconnecting to a peer already present in the list. In contrast to other epidemic protocols such as Newscast, we use both exploitation and exploration branches, we limit the randomness of peer selection during the exploration, and we implicitly cluster peers into (trusted) social groups [17].

5.5 Collaborative Downloads

In Collaborative downloads, each peer has to take one of two roles; either a Collector or a helper. The collector is the peer that needs the complete file and in order to improve the download performance , the collector peer collaborates with group of other peers called the helpers. These helpers download pieces from various sources and sends those pieces to the collector. A helper peer doesn't seek anything in return from the collector, give more priority to collector requests and show no interest in the content the collector is downloading. The collector selects the best source from the helpers thus optimizing its download performance.

The fairness of bandwidth sharing is enforced by a hybrid approach, as the bandwidth is accounted differently between non-collaborating peers, and between a collector and its helpers. Peers that are not members of the same collaboration can exchange chunks with the standard bittorrent's tit-for-tat algorithm. The BitTorrent tit-for-tat bartering strategy ensures that for a peer the amount of incoming data is roughly equal to the amount of outgoing data. The tit-for-tat mechanism guarantees fairness only within a single download session (download of one file). The asymmetric nature of the collector-helper relation implies that the bandwidth contributed by the helper to obtain the chunks

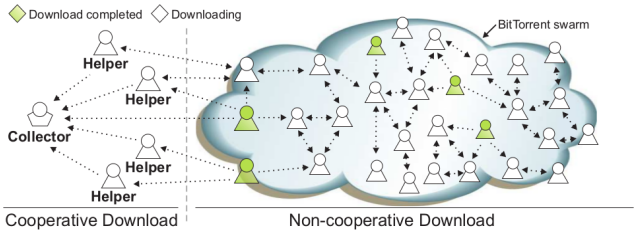


Figure 4. Overview of cooperative downloading, Source:[17]

Figure 5.

for the collector in the current download can be reclaimed only during later downloads. We base the fairness between collaborating peers on a notion of a promise. A collector recruits helpers by giving them a promise that the bandwidth invested in boosting the collectors download performance will be returned in the future. Helpers contribute their currently idle bandwidth knowing that later the roles may be reversed a helper will become a collector and the current collector will pay back for the consumed bandwidth by participating in the download as a helper. The role of a promise is to provide an incentive for collaboration. This role can be fulfilled only if the system provides guarantees that the promise will be delivered. [20] [17]

6 Conclusion

In this paper, we discussed the operation of bit-torrent, the framework behind its working nature and have listed the proposed modifications to deal with performance issues , research challenges and factors to be considered while implementing bit-torrent in a P2P social network. Bit torrent has been slowly migrating from its conventional public file sharing role to a social network. In todays world, recognition of having a more distributed and decentralized social network has grown largely among people. We have presented a paradigm for the design of bit-torrent style sharing feature in P2P social network. Exploiting the feasibility of having a complete de-centralized robust torrent feature has not been deployed officially and is open to further research.

References

[1] M. Defeche, "Measuring IPv6 Traffic in BitTorrent Networks," in *IPv6 Operations*, 2012.

[2] Ernesto, "BitTorrent Still Dominates Global Internet Traffic, http://torrentfreak.com/bittorrent-still-dominates-global-internet-traffic-101026,(last checked: 23.05.2013)),"

[3] K. B. Peng Shi, "MegaTorrent: An Incentive-Based Solution to Freeriding in P2P File-Sharing Networks,"

[4] R. Izhak-Ratzin, "Improving the BitTorrent Protocol Using Different Incentive Techniques," 2010.

[5] B. Cohen, "Incentives Build Robustness in BitTorrent," 2003.

[6] X. Y. Marios Iliofotou, Georgos Siganos and P. Rodriguez, "Comparing BitTorrent Clients in the Wild:The Case of Download Speed," in *ACM Computing Surveys(CSUR)*.

[7] "Vuzu:The Bit Torrent Client." http://www.vuzu.com.

[8] C. Gong, W. Chunming, and J. Ming, "Region tracker: An effective way to localize bittorrent traffic," in *Wireless Communications Networking and Mobile Computing (WiCOM)*, 2010.

[9] M. k. R. Charles P. Fry, "Really truly Trackerless Bit Torrent," 2006.

[10] R. L. Xia and J. K. Muppala, "A Survey of BitTorrent Performance," in *IEEE Communications surveys*, 2005.

[11] R. Thommes and M. Coates, "BitTorrent Fairness: Analysis and Improvements,"

[12] G. U. Arnaud Legout, Pietro Michiardi, "Understanding BitTorrent: An Experimental Perspective," 2005.

[13] I. R. Kyuyong Shin, Douglas S. Reeves, "Treat-Before-Trick : Free-riding Prevention for BitTorrent-likePeer-to-Peer Networks," in *Proceedings of ACM conference on Electronic commerce,ACM*, 2005.

[14] E. K. L. Z. Arnaud Legout, Nikitas Liogkas, "Clustering and Sharing Incentives in BitTorrent Systems,"

[15] W. Galuba, "Friend-to-Friend Computing:Building the Social Web at the Internet Edges,"

[16] J. E. J.Pouwelse, P.Garbacki and Sips, "The Bittorrent P2P file-sharing system: Measurements and analysis," in *International Workshop on Peer-to-Peer Systems(IPTPS)*, 2005.

[17] W. B. Y. I. E. R. v. S. S. J.Pouwelse, P. Garbacki, "Tribler: A Social-Based Peer-to-Peer System," in *International Workshop on Peer-to-Peer Systems (IPTPS)*.

[18] "The Tribler Client,www.tribler.org/trac/wiki,(last checked: 25.05.2013)),"

[19] R. Vliegendhart, "2-Hop TorrentSmell-A Distributed Tracking Algorithm," in *Tribler Documentation*, 2009.

[20] P. Garbacki, A. Iosup, D. Epema, and M. van Steen, "2Fast:Collaborative Downloads in P2P Networks," 2006.

Handling overlay attacks in Pastry

Christopher Petri
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email: cpetri@upb.de

Abstract

Pastry is an overlay to crate self-organizing P2P networks. Upon the Internet Pasty forms a Network with own namespace and routing protocols. The Network is decentralized, so there is no central authority. This kind of network allows peers to join and leave rapidly, so these kind of network can have a high turnover. Every node joining the network connects to other nodes. They build the network by exchanging information about other nodes routing messages between all nodes.

P2P networks can expand to a large scale and that for it is necessary to think about possible attacks against the network. Not every node which wants to join the network will cooperate with the other nodes. There might be attackers join to the network to harm it.

This Paper provides an overview of some possible attacks against P2P networks and espacialy Pastry networks with a description how they operate and how they impact. For each attack at least one countermeasure is given with a functional description. Thereby attacks with one and many nodes will be considered.

1 Introduction

Distributed applications do not work in the principle of client and server architecture. In P2P networks each participant acts as client and server at the same time. So they are preferred when many people need a direct communication with each other. Some possible types of uses are content distribution, file sharing and communication. Somehow all of these could be useful in an social network, there is scientific research to build social networks upon P2P networks. Using a P2P network to establish a social network could make it possible to keep control of private data while distributing data over the network.

Pasty builds such an network up on the internet. The infrastructure of the internet is used to build a P2P network without changing the infrastructure. P2P networks work with nodes. Nodes connecting to other nodes under consideration of rules are able to form a network. In Pastry these Nodes form a ring, in which they forward messages from other nodes to other nodes. This way every participant can communicate with any other. The forwarding of messages must follow rules to ensure the reliability of the system.

In a P2P network a social network could be represented. Each participant or profile gets a own node. The communication of the nodes can be imagined like writing notices in school. Albert wants to ask Bertha something. He writes a note and puts Berthas name on it and gives the note to someone who is closer to Bertha than himself. If every one give the note to someone who is closer to Bertha than himself, the chance is high, that Bertha receives the note.

In difference to a class in a P2P there might be someone who wants to join the network, but does not want do participate on the common benefit by working together with the other nodes. Someone even might only join the network to harm or destroy it.

P2P networks can grow quite large in the number of participants. It would be careless to assume that everyone will follow the rules. So there is a need for security in P2P networks.

There already has been research to find different potential ways to attack P2P networks and how to defend these attacks. The scope begins with a single node disturbing the communication to a mass of nodes trying to destruct the whole network.

The following section will describe how a network built with Pastry is organized and which structures are needed and build by the network. Within this section is a description how a node can participate in the network. Section 3 tells about the some of the different types of attacks that are especially for P2P networks. After each attack a counteraction is given that may be sufficient to handle the attack.

2 Forming a network with pastry

Pastry forms an structured overlay network. This means, it uses an existing network to build a new functionality up on it. Therefor, it uses its own namespace and identifiers. To be able to use the functionality of the underlying net-

work it is necessary to be able to translate namespace and identifiers between the two layers. In Pastry distributed hash tables (DHT) are used for that demand. So each node in the network has a own table with IPs and identifiers of their own namespace. For general information about structured P2P networks [1] and [8] could be useful.

Each node in Pastry needs an unique identifier. This identifier is called the nodeID and it has 128-bit length. There are different ways to assign a nodeID to a node, each way with different aspects to security. Pastry only demands that the nodeIDs are distributed over the whole namespace. This is necessary, because each live node will be in charge for the region of its nodeID up to node with the next higher nodeID. Without distribution across the namespace one node might get in charge of a much larger region than should for security reasons or it could handle because of technical issues.

To know the region a node is in charge of, the node have to know its neighbors. For these demand it would be sufficient to just to know the neighbor with the next higher nodeID. But when these neighbor leaves the network it is more complicated to get to know the new neighbor with the next higher nodeID than to keep track of a neighborhood set. The neighborhood set contains the nodes with the $l/2$ numerically closest larger and $l/2$ numerically closest smaller node IDs. This way, even one node leaves, a node still knows its direct neighbor. The nodes with the highest nodeIDs in the namespace and the nodes with the lowermost nodeIDs also keep track of each other in the neighborhood set. This way Pastry forms a circular namespace with the possibility of bidirectional routing.

For routing in Pasty, the 128-bit identifiers get logically divided into 2^b (b is a configuration parameter with a typical value of 4) bit parts. Each node has a routing table with $128/2^b$ rows and 2^b columns. All entries in the row i of a node share the first i bit parts (or $2^b * i$ bits) with the actual node. For each possible $i + 1$ bit part, different from the current nodes $i + 1$ bit part, exits a row, so that there can be an entry of a nodeID which matches these first $i + 1$ bit parts. There can be many nodes fulfilling this criteria in the network, depending on the population of the network. Which one is chosen depends on the used metrics.

When a node forwards a message it first compares its own id with the destination of the message. The node will then chose a node which shares at least one bit part more with the destination then himself from the routing table. If there is no such node in the routing table of the current node, the message will be forwarded to a node witch shares the same amount of bit parts with the destination like the current node does, but has a numerically closer nodeID than the current node. Such a node must exist (unless at least $l/2$ nodes in the neighborhood set fail) in the neighborhood set of the current node with one exception.

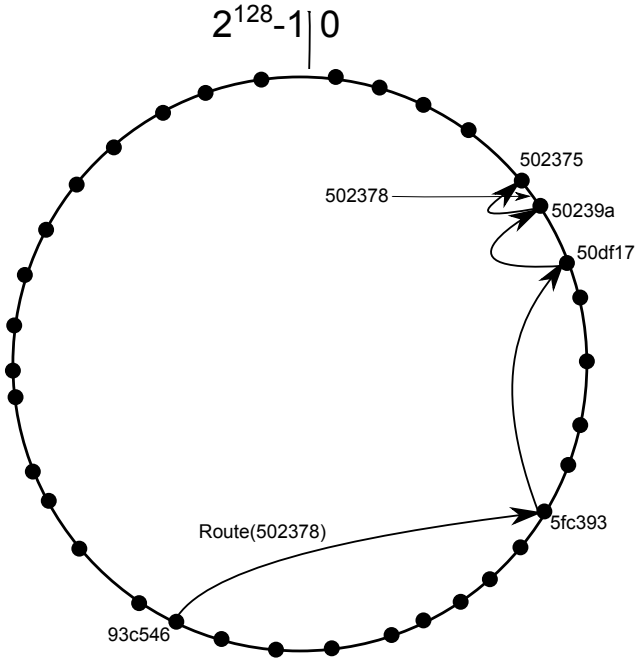


Figure 1. PastryRing

As said before, a node is in charge of a region of the namespace. If the destination of a message is larger than the nodes nodeID but smaller than the numerically next larger nodeID of the next live node, a node will accept the message as its destination.

In 1 you can see a ring formed by Pastry nodes. In this example b has the value 4. Node $93c546$ sends a message to 502378. Node $93c546$ looks up a node from its routing table 2 with the first b-bits closer to the destination then itself. In this example this would be the node $5fc393$. Node $5fc393$ now looks up a node with b more bits equal to the destination then itself and so looks for an appropriate node in its routing table and so forwards the message to $580df17$. $580df17$ forwards to $50239a$ in the same way. $50239a$ is a node direct beside the destination address, but its own nodeID is larger than the destination address. So the node with the largest numerically smaler nodeID is responsible for the message. In this example, this is node 502375. The figures are created in backing of [8].

To be able to fulfill routing tasks the network must dynamically maintenance itself. This means for live every node, that it must keep its routing table and neighborhood set up to date. No matter other nodes leaving or joining.

2.1 A new node joins the network

The most important thing for each node is its nodeID. As already said, there may be different ways of obtaining with different aspects to security issues. Pastry only demands that the nodeIDs must be distributed over the namespace. This would be possible if each node is allowed to create

0x	1x	2x	3x	4x	5x	6x	7x	8x		ax	bx	cx	dx	ex	fx
90x	91x	92x		94x	95x	96x	97x	98x	99x	9ax	9bx	9cx	9dx	9ex	9fx
930x	931x	932x	933x	934x	935x	936x	937x	938x	939x	93ax	93bx		93dx	93ex	93fx
9330x	9331x	9332x	9333x	9334x		9336x	9337x	9338x	9339x	933ax	933bx	933cx	933dx	933ex	933fx

Figure 2. PastryRoutingTable

its own nodeID freely by random as long as its nodeID is unique. For security issues this would be the worst way of assigning nodeIDs, but later to that.

Lets assume a node A has a unique nodeID and wants to join the network. The node A needs an access point. This will be an other node X, that is already in the network. A must have the IP to be able to communicate with node X. If this is given, the new node A will be able to send a special message to X. The message will be routed to the node with the numerically closest smaller nodeID Z. This will happen by routing the message over X with the nodeID of A as destination. By Z having the numerically closest smaller nodeID, it is in charge of the region where the message was send and A wants to join. A will gather the routing information of the path from X to Z for its routing table.

Z will respond to A with its own neighborhood set. This will be used as the initial neighborhood set of A. Now A introduces it self to his new neighbors which means it causes an update process where the neighbors update their neighborhood sets and routing tables. Now X is a member of the network and in charge of a part of the namespace region Z was in charge of.

In figure 1 you can see node X sending a message to node Z via node X. The message is send to the nodeID of X. Befor X is full integrated in the network, Z is responsible for the namespace where the nodeID of X is included. So node Z will be the receiving node of the message.

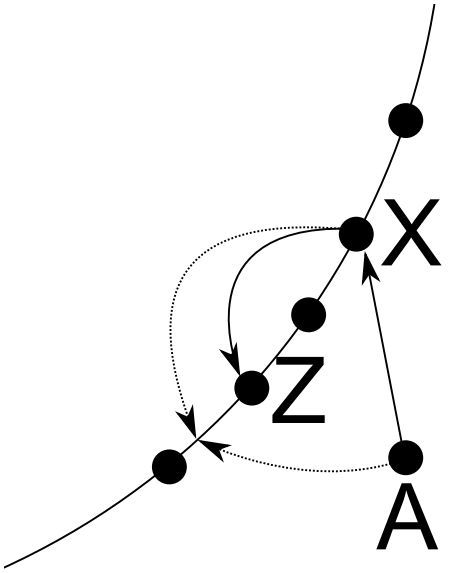


Figure 3. Joining

The update process of the routing tables and neighborhood sets can be done by a small number of messages $O(log_2^b N)$ between a small number of nodes. The same way, the notes can react to a abrupt node failure, to establish a stable state.

3 Attacks and countermeasures

As the network grows, it get more and more likely that there will be someone who misbehaves. Maybe even not by purpose, but even one malicious node could bother the communication. When someone tries to disrupt interaction it just can get worse. In the following sections some attacks that are especially against P2P systems are described and how they can be encountered.

3.1 Incorrect lookup routing

When a node A is trying to get in touch with another node Z, it will send messages to the nodeID by forwarding it to B, the next hop on the route to Z. Correct nodes will forward the message as described and form a route from A to Z on which the message is transported. A will know that Z got the message on response of Z.

On the route from A to Z may be a single malicious node that does not forward the message to the correct node. It could be possible the malicious node X forward the message to a wrong note or a none existing node. If the malicious node belongs to an adversary it is also possible X drops the message on purpose.

After a timeout A must suppose that the message did not reach Z, because A got no acknowledgment. So A will try

again sending the message to Z by forwarding it to the next node B. While there is no change in the routing tables, all correct nodes will behave like before thus the message will be reached the malicious node X again. So retransmission of the message will fail while there is no change in the routing table.

The malicious node X might not be removed from the routing tables as long as it is not dead and response to other nodes. So an update of the routing tables might not solve the problem.

One easy way, of establishing a connection between A and Z, is to reroute the message. Node A could choose node C instead of node B to forward the message to, if C is also closer to Z than A. This would not be a bit better, if C forwards to the same node as B did or both routes join on another node before X. So a way of identifying the malicious node may be useful.

A possible solution could be what [10] called iterative routing. Node A would chose again node B as next step, but instead of forwarding the message, A asks B for the next step on the route to Z. Then A asks the node responded by B for the next and so on. When the malicious node X get asked for the next step it might not answer or answer with a next hob, that can't be correct because of the routing rules or does not answer. Then A knows that X does not work correctly an can ask X predecessor on the route for an alternative.

Because of the high latency of this way of message routing [10] proposed another solution the called tracer routing. In this strategy the message is again forwarded directly on the route, but each node the message passes must tell A to which node they forward the message. This way A get the routing information needed to avoid a malicious node, but does not need to wait for each node to answer till it can talk to the next one.

In 4 you can see the described routing methods. The figure is created in backing of [10].

3.2 Incorrect routing updates

For joining and leaving nodes, routing table updates are necessary. In an network with high turnover rate or short time participating nodes they could happen quite frequent. As the fundamental function to keep the network operating, it is unsurprising that this functionality is target of adversaries.

Every node constructs its routing table out of information it gains from other nodes. A malicious node may try to corrupt the routing table of a node, by trying to send wrong updates. That way an adversary may try to use an innocent node to redirect messages to other malicious or non existing nodes. These kind of attack is also called routing table poisoning.

The incorrect routing updates are quite simple to expose. Pastry requires a correct prefix for each table entry. If this

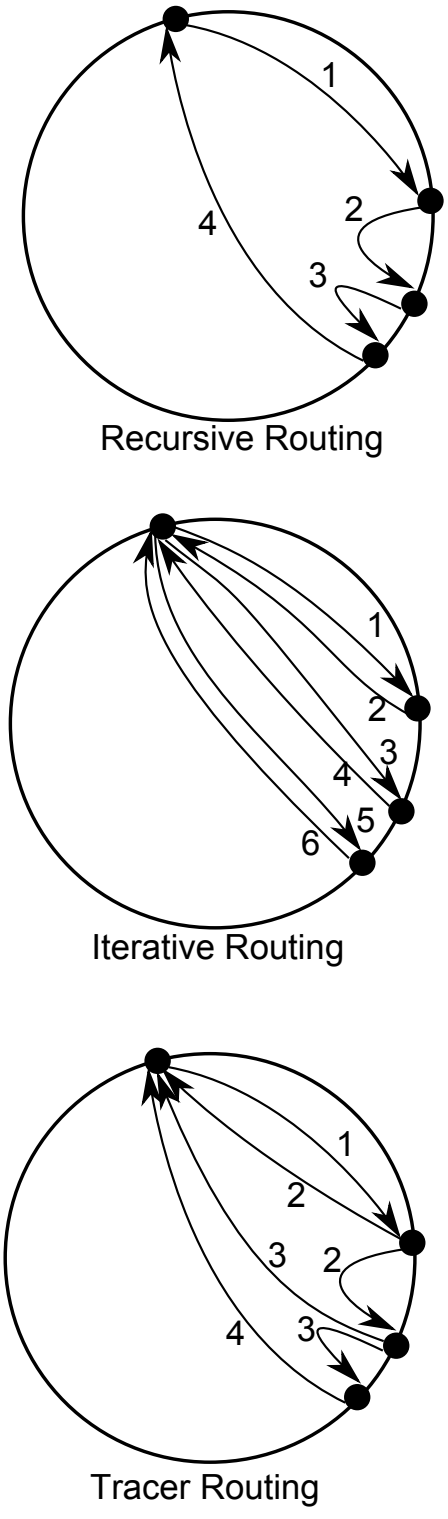


Figure 4. RoutingMethods

requirement is violated, the whole table can be ignored. But even if all formal requirements are fulfilled, a node can at least test the reachability of a node, out of the update table, to ensure it is alive. Also a node could test if a node routes correctly to a target by checking the routing information of a message.

3.3 Join-Leave-Attack

Every node needs to join the network by contacting a node within the network. The joining node gets information from the nodes in its neighborhood and causes routing updates within the neighborhood. A leaving nodes also creates a need for routing updates to reestablish a stabilized state. An attacker might join and leave the network repetedly and rapid to cause many routing updates within the neighboring nodes.

Within a P2P network every nodes is able to leave without any forecast. Every time nodes are able to join the network. So you can see that the construction of the network must be made this way, that a Join-Leave-Attack is somehow nothing special to this kind of network and a basic condition.

3.4 Partition

As described in [6], a node is very vulnerable when joining the network. To join a network, a node has to contact some node that is already participant of the network. The new node gets the first information of the other nodes in the network from the first node it contacts. At this time it is possible for the contacted node to maneuver the new node into a parallel incorrect network. This network could run the same protocols and act like the real network. The parrallel network could also have data from the real network or being cross-registered with the real network so that the new node does not notice it has been partitioned into another network. With cross-registered nodes it is also possible the the new node is in company with other nodes which got hijacked before what would make the parallel network more authentic.

This kind of attack could be used to gather information from specific nodes or functionality of the real network. To prevent this kind of attack, a node could talk to other known nodes from the past. By crosschecking the view both nodes have on the network, it is possible to identify, if they are in the same network. If the node joins the network for the first time it must have some kind of trusted source where it could enter the network safely. Otherwise it could be possible that a hijacked node would identify the real network as a parallel network, because it only knows the parallel network.

3.5 Dos and DDos

As in any other network a Denial of Service (DoS) attack in a P2P network tries to flood the target with request. The incoming queue get stuffed with requests, so that the target can not receive real requests from correct nodes. A Distributed DoS (DDos) attack is mostly the same, but uses multiple hosts to perform the flooding of a target.

In [9] there is mentioned a special kind of DoS/DDoS attack called Query Flooding Attack. One or many nodes sending query to other nodes with the IP of the target. The querried nodes response to the target. This way a node can get flooded with many acknowledgments from many nodes with many different IPs. The countermeasures against the common query flooding attack can not oppose these variant because the whole network can get abused for this attack.

3.6 Sybil attack

The Sybil attack uses the fact, that in a distributed system, a node experience other nodes as independent entities even if they arent. Without any countermeasures an adversary is able to insert multiple logical nodes into the network by using only one physical entity. The Sybil attack does not harm the network itself but can be used to do this. An adversary could flood the network with his nodes to dominate it. Then the adversary could destroy security and redundancy mechanisms, gather secured information and partition single notes and even the network.

The conclusion of [3] is, that there can not be any fully trusted relation between two nodes in a P2P network without a fully trusted central authority (CA) that can ensure the logical and physical identity of a node. To do that, secure communication would be necessary.

In one approach the CA is responsible for handing out private keys to nodes who want to join the network. Then the node can calculate its corresponding public key, to use it as nodeID an join the network. Since the CAs public key is known to all nodes and the CA knows the private key of every node, each node can ask the CA if a other node is who it claims to be. To verify a node and get a private key, one need to authorize itself with some already existing trusted and reliable system, for example via credit card.

In this approach the CA limits Access to the network by IP and IP ranges by binding them to the private key. It is necessary to allow multiple nodes for one corresponding IP, otherwise users behind NAT firewalls would be excluded. On the other side, adversary with access to an IP range, like one of a big company, should be limited in his capabilities to insert new nodes into the network. If an adversary has access to a botnet with many different IPs this is access control is not effective anymore.

There are also different ways of preventing sybil attacks without use of an CA, like in [2] where a group of nodes is able to register a new node. Another approach [7] uses the P2P environment to let nodes authorize themselves. They want to create self-certifying data out of IP, MAC and other measurable values which can be tested by other nodes. So far this does not prevent a sybil attack as each node of a sybil attack may be able to authorize itself with the correct

data. To limit the number of nodes they use computational puzzles to be solved. These puzzles require much computation time, so that an adversary is not able to solve many of them an so limit the number of nodes. If an adversary has access to a botnet or many computes of a company this approach might not be quite sufficient. Another Problem is that the measurable values used to certify may change over the time.

A very interesting approach are SybilGuard an SybilLimit out of [11] and [12] which base on properties of social networks. They look at a social network as a graph with nodes as persons and edges as real and trusted human relationships like friendships. The sybil nodes will not be able to create or take as much of these edges to human and honest nodes as another honest and real human can create. For that reason the graph will split into two parts, one with hones and one with sybil nodes, connected by only a few so called attack edges. In figure 5 an example demonstrates the connection of hones and Sybil nodes via attack Edges. The goal is to identify these attack edges so that one is able to identify if another node is in the same region or not. To identify these edges directly a view of the whole network would be necessary. Even if every node had this overview over the whole network, the problem to find these cuts is NP-hard. Instead of searching directly for these edges, they use random walks on the graph. The walks are so designed that two nodes can check if they have walks in common and so identify with high probability if they are in the same region of the graph. This approach seem to have good results against the sybil attack, but because of the need of a social network this is a very special approach and will not be useful in most of the P2P networks.

3.7 Eclipse attack

Within an eclipse attack, malicious nodes want to get into the neighborhood set of correct nodes in large number so that they get the possibility to eclipse nodes. In other words, they want to block all routes between two sets of nodes. The eclipse attack can be used to disturb routing functions in the whole network or, what is more effective, against single nodes or a small number of them, to separate them from the network. The malicious nodes may use routing table poisoning to position other malicious nodes in the network. This way, they can increase their fraction in the routing tables with each update. They may also use sybil attack to get into the network in large number. An considerable eclipse attack in two regions of the network may also be able to partition the network.

One countermeasure against eclipse attack is to use two routing tables [4]. One routing table, that is used like normal, called the optimized routing table. This table will be used to find potentially reliable nodes and generate routing entries with respect to proximity and other metrics to find an optimized route. The other routing table, called verified

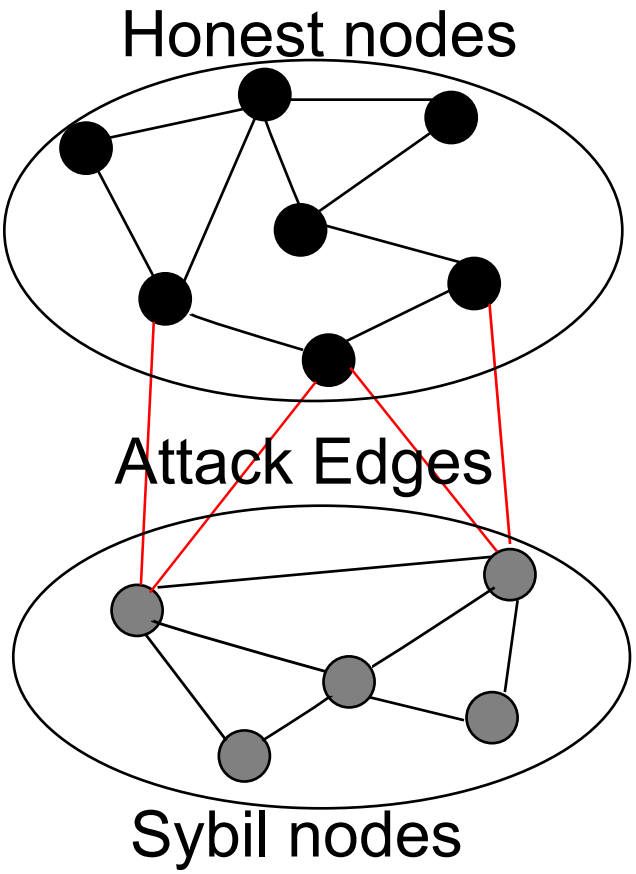


Figure 5. Sybil

routing table, do not care for else other then the trustworthiness of node. Only nodes than can be verified will be registered. As long es everything works fine, only the optimized routing table will be used. If failures occur, and the route got tested to fail, the verified routing table get used. This way the optimized routing can get used untill it fails. In this approach only one of two routing tables can get poisoned by the eclipse attack. This way there is always a back up, but the optimized routing table get compromised without the chance to detox on another way than the normal routing updates which are to blame that the optimized routing table is poisoned. The biggest problem of this approach is, that a singe malicious node in the verified routing table could violate the whole concept.

The in-degree of a eclipse node must be much higher than the average in-degree of correct nodes, during an eclipse attack. In [5] this observation is used to build a defense against eclipse attacks. A node should not add a node to its routing table, if it has a high in-degree and therefore is a eclipse node with high probability. So the in-degree of a node get bounded. To prevent eclipse nodes placing themselves in front of their target, also the out-degree get bounded. A node A hast to verify its companions repeatedly. To verify a node X, the routing table and a backpointer list is needed. The backpointer list contains nodes, which are routing towards X. To verify the correctness of these list, node A uses an anonymizer V to request these list. The anonymizer node V should be randomly chosen out of the neighbors of the node X. That way X can not create a special version of its routing and backpointer list to betray A. If V and X are working together, then they would be able to create a special version of the list to betray A, but this case is easy to detect, if A ask more than one anonymizer at different times for the routing and backpointer list of X and compares them. If a list does not fulfill the degree-threshold A drops X. If A is not on the list of X, A also drops X. At this point X could still be a malicious node that just had a lucky guess and put A on the list. So A has to ask other nodes on the list if they are relay working together with X, if not X get dropped by A. This approach seem be effective to prevent that nodes can be eclipsed because the nodes use more different routes but there still can be a large amount of eclipse nodes disrupting messages.

In figure 6 four malicious nodes eclipsed a node by overtacking its neighbourhoodset. This nodes can not send messages to other nodes without forwarding the message to a malicous node.

4 Conclusion

In the previous section we have seen some serious threads against the security of P2P networks. The decentralized nature of the network makes it vulnerable to some attacks, that are only a threat to these kinds of networks. But

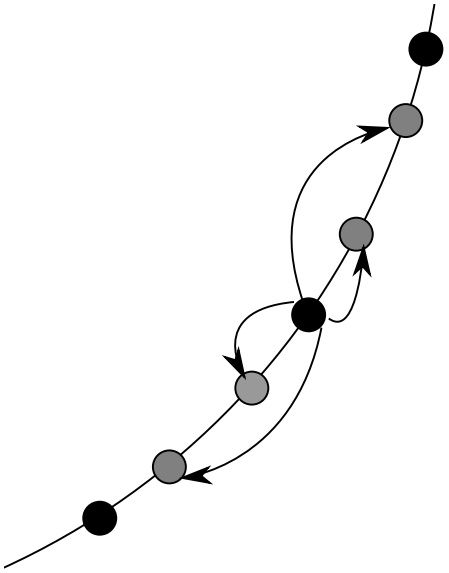


Figure 6. Eclipse

there are also some countermeasures with some promising results against these attacks.

As you have seen, if it comes to an attack with many organized nodes, these countermeasures are not able to prevent the attendance of these malicious nodes in the network completely but they are able to prevent major damages to the network.

It is clear that every countermeasure needs extra effort. This can be an longer latency, a special central authentication unit and even a whole social network. What is really necessary is a way to establish secure communication. Since there is still no entirely satisfying solution to that problem, more research could be helpful.

So far there are multiple solutions to defend each attack with more or less disadvantages. The presented countermeasures of this paper show that is possible to build a social network upon a P2P network with a practically acceptable security.

References

[1] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach. Secure Routing for Structured Peer-to-Peer Overlay Networks. In *OSDI '02: Proceedings of the Symposium on Operating Systems, Design and Implementation*. ACM, 2002.

[2] J. Dinger and H. Hartenstein. Defending the Sybil Attack in Peer-to-Peer Networks: Taxonomy, Challenges and a Proposal for Self-Registration. In *ARES '06: International Conference on Availability, Reliability and Security*. IEEE, 2006.

[3] J. Douceur. The Sybil Attack. In *IPTPS '01: Revised Papers from the first International Workshop on Peer-to-Peer Systems*. Springer, 2002.

[4] A. Singh, M. Castro, P. Druschel, and A. Rowstron. Defending Against Eclipse Attacks on Overlay Networks. In *SIGOPS '04: Proceedings of the European Workshop of the Special Interest Group on Operating Systems*. ACM, 2004.

[5] A. Singh, T.-W. Ngan, P. Druschel, and D. Wallach. Eclipse Attacks on Overlay Networks: Threats and Defenses. In *INFOCOM '06: Proceedings of the IEEE International Conference on Computer Communications*. IEEE, 2006.

[6] E. Sit and R. Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. In *IPTPS '01: Proceedings of the International Workshop on Peer-to-Peer Systems*. Springer, 2002.

[7] G. Urdaneta, G. Pierre, and M. van Steen. A Survey of DHT Security Techniques. In *ACM Computing Surveys*. ACM, 2011.

[8] D. Wallach. A Survey of Peer-to-Peer Security Issues. In *ISSS '02: Proceedings of the International Conference on Software Ssecurity: Theories and Systems*. Springer, 2002.

[9] L. Wang. Attacks Against Peer-to-peer Networks and Countermeasures. In *Seminar on Network Security at TKK*, 2006.

[10] X. Xiang and T. Jin. Efficient Secure Message Routing for Structured Peer-to-Peer Systems. In *NSWCTC '09: Proceedings of the International Conference on Networks Security, Wireless Communications and Trusted Computing*. IEEE, 2009.

[11] H. Yu, M. Kaminsky, P. Gibbons, and A. Flasman. SybilGuard: Defending Against Sybil Attacks via social Networks. In *SIGCOMM '06: International Conference of the Special Interest Group on Data Communication*. ACM, 2006.

[12] H. Yu, M. Kaminsky, P. Gibbons, and F. Xiao. SybilLimit: A Near-Optimal Social Network Defense aginst Sybil Attacks. In *SP '08: Proceedings of the Symposium on Security and Privacy*. IEEE, 2008.

Survey on E-Learning and social collaboration systems

Siddharth Kaneriya
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email: Siddharth.Kaneriya@mail.upb.de

Abstract

In today's world, use of social networks has increased rapidly. Moreover, content and data sharing are one of the most crucial requirements in the current era. Sharing of content is a broad category in itself and leads to many platforms like Facebook and LinkedIn provides capability to connect users all over the world and interact and share content. On the other hand, platforms like Jive and SharePoint provides a collaborative platform for sharing content on an intranet level and could be equally useful for company to communicate or for an university to provide a common learning platform. The goal of this paper is firstly to understand such solutions in depth. Further, this paper argue about the workload specific parameters like user count and usage patterns while utilizing above solutions. Lastly, this paper should enable a reader to differentiate between the present solutions under the collaborative and e-learning platform with the help of their workload requirements.

1 Introduction

1.1 Outline

In recent years the trend of working has completely changed because of the high competitive environment produced by the market. It seems to be getting more and more complex as it includes many individuals working on the same goal. Different people have different styles of working and different perspectives to see and analyze the same thing. Not only working trend but decision making process, scheduling process, rule management and project management have also become very complex and complicated due to the large number of people working on the same goal. On the other side, the trend in academia is also changing very frequently due to innovations in the social media and advanced technology. Computer and social network has become a very essential part of a normal human routine life and there is hardly any field in the world, which does not use networks or computer. Social networks were perceived to be the solution for such problems, but due to their fixed and rigid structure, they cannot be used for specific organization. While *collaborative system*, which is also called as *collaborative software*, *social software* or *collaborative network* is a solution for such problems. It provides a common platform, where people can work together and share their views with the other people who are involved in the same task. While in the academia, it can be used to provide the distance learning and e-learning by providing educational tools like discussion forum, wikis, online tutorials and training etc.

This section briefly describes *online social networks* and *collaborative social intranets*. It also argues, why online social networks can not be used for the standard organization because of their structure and behavior.

Millions of normal users, business officials and students in the world are a part of social networks like Facebook, LinkedIn, Twitter, which depicts very huge amount of usage of social networks among the people [21] . Moreover, social media has been gaining recognition as contributing a sustainable competitive advantage to corporate sectors and other sectors like administration, human resource and academia. Social networks are very important part of the social media and provide two basic facilities [15] .

1. **Content Sharing**: It includes file sharing like video, pictures, articles and documents. e.g Video sharing gave a birth to the new technologies called e-learning or e-lectures.

2. **Connectivity**: It connects users from all around the world by bringing all the users on a common platform.

Such facilities enable users to communicate with other users and that too with a glance of one click. It also allows users to share their personal information with other users within a group or on a particular forum. This is a main reason, why social networks are mainstream in current time [14] . Use of social networks are not just limited to personal level as businesses are recognizing social networks as the new means of promoting their products and services. Software giants like Microsoft, IBM have discovered that it is very crucial to

have such social networks for business organizations. As a result we have Sharepoint, JIVE SBS, Google apps etc.

1.2 Online Social Networks

"Online social networks are web based services, which allows users/actors/individuals to create public or semi public profile within the bounded system, articulate a list of other users with whom they share a connection, view and traverse their list of connection and those made by other systems. A relation between these connections varies from system to system" [9]. It is very clear from the definition of a social networks that they allow content sharing and information sharing as discussed above. Most of the social networking websites offer same functionality and use centralized server to store user information. Recently social networking websites have started offering a wide range of technical features like online marketing, adverts and promotions of the business products and services etc, which has given rise to new avatar of the social networking. But as we know, new features always bring new challenges. As a result, we have new challenges like privacy of contents, security of important data and many other challenges as most of social networks use the centralized server to store data and information. Moreover, their availability is not just limited to particular organizations or particular users, they are meant to be for everyone. Because of this phenomenon of social networks, they can not be used for a larger firm or business organization. Besides, these social networks are not application oriented because of their fixed structure and rigid functionality.

1.3 E-learning and Social Collaborative Networks

"Collaborative software or collaborative social software are integrated software which provides different functionality for decision making, workforce sharing, calendaring the events and organizing and finalizing the goals, which are used by the virtual organizations, virtual enterprises, highly integrated and dynamic supply chains in order to increase their productivity, to co-ordinate and schedule the workforce and to achieve the common goals by prevailing the lone potentiality" [7]. There are many solutions available in the market. For example JIVE SBS, Atlassian, Sharepoint etc. Advanced modification into the social media and the popularity of social networks among normal users and business organizations has given rise to social collaborative system. Earlier business organizations used to use emails mainly as a medium of communication, although, it is consistent method for the office communication but with some advanced modification and integration with the collaborative software. For the decision making, they used to use online poll or manual poll, which was a time consuming and lengthy process. Collaborative software has made such procedures very easy by enabling the involved entities to participate into the voting procedure and sharing the important knowledge [17]. For this purpose, we use collaborative decision making tools integrated with collaborative software. Moreover, the application of collaborative social software is not just limited to increase sales funnel or productivity. We can easily integrate such software with our existing work software. e.g integrating collaborative software with e-learning platform gives rise to a system called *computer supported collaborative learning* [16], which can be very useful in the academia and distance education. Moreover, it can also be a good solution to an administrative software in academia and training institutes, which indirectly serves a good purpose for the human resource industries.

1.4 Comparisons

Goals and objectives of social networks and collaborative software are totally different. On the other side, functionality provided by them, is almost consistent on various level but they are very different in terms of behavior. Collaborative software provides application specific functionality while online social networks do not provide such functionality. Sharing same fashion and functionality does not mean that both the networks behave in a same manner. Social networks are more of general purpose software, where people share their views, contact other people, hangout with other people, who share common interests and also share the multimedia. All these activities of social networks are stored in the centralized server of the service provider. Because of this phenomenon of social networks, sometimes people face some severe problems like lose of personal data, misuse of the identity and many other problems [12] . Such issues become more critical when we think of using social networks for a particular group or organizations because of a fear of losing confidential data. Also it is not suitable for organizations, which have their own standard ways of working because of its fixed design structure and limited functions [18] . Collaborative software/Social collaborative software is more like an application or product specific software unlike social networks, with more specific functionality to offer to users. Collaborative software provides a common platform to the people who work to achieve the same goal or who work on the same project by providing a platform for communication, content sharing and decision making functionality. Moreover, collaborative software uses the internal server of particular organization in which it is deployed, so there is no question of security and lost of the data. Agile structure and behavior of the collaboration software makes it more superior than social networks when it comes to internal communication. In the next section we will discuss some current collaborative software, their behavior and functionality offered by them.

2 Survey on Current solutions available in the market

As we discussed in introduction the main aim of social collaboration system is to enable the effective sharing of data and information so that all involved entities get benefit from it. It also provides support to mange and co-ordinate the entities by providing effective communication and sharing platform irrespective of geographical location. Due to such qualities, it gave a new meaning to global market and commodity sectors. Besides, it is also gaining lots of importance in the other sectors like academia, Human-Resources, Hospitals etc.

Currently, there are many collaboration solutions available in the market, so choosing the right collaborative tool for the organization has become a very crucial as effective collaboration is must in order to gain the benefits from it. In the current section, we will discuss how such system works and the approaches for social collaborative system by briefly discussing the current collaborative solutions available in the market.

2.1 JIVE Social Business Software

JIVE social business software(JIVE SBS) is a revolutionary social collaborative software which provides different platforms like communications platform, collaborative platform, social platform to assist modern business organization to enable their mass productivity and also helping them to gain more immunity to survive in this competitive market. Moreover, it can be easily deployed with the existing cloud or on premises. Such qualities of JIVE SBS make it organization friendly and easily adoptable into the working environment [3]. Being a leading collaborative system, JIVE SBS provides the organization wide easy communication by creation and allocation of the discussion forums and specialized blog areas [3]. It also supports review system of such blogs and forums by enabling the comment and report facility so that anybody can share their views and provide the innovative ideas in order to make projects more effective and successful [4]. JIVE SBS is also armed with various collaboration functions and provides a platform for the sharing of the knowledge base on different levels. Main key feature which makes JIVE SBS even stronger is a facility to add people to the forum who are not part of a group or a company, so that they can help the employees by providing their expert opinion. Besides it offers an option called live search which helps users to get external help and also users can set up their own profile and own dash board in JIVE SBS and they can also edit their blogs post etc. Such phenomenon of JIVE SBS makes it very user friendly and organization friendly. JIVE SBS also provides *unproblematic clustering* [4] for big companies and easy integration with existing tools like Microsoft outlook, office and SharePoint(which itself a collaborative software) [3]. Although, JIVE SBS is a corporate software, it provides a very strong platform for the social networking which helps users to maintain their biographic data and also help them to stay in touch with other users. JIVE SBS is very agile software, which can be deployed anywhere. For example Jive SBS can also be used as a service and it can also be used directly by deploying it into the internal company server or cloud(on premises). JIVE SBS supports various project management tools and it supports various data providers [4]. Moreover, the administration of the JIVE SBS is also very easy and uncomplicated as it provides security on each and every level and administrators can make changes on each level. From the case studies of the engaged companies with JIVE SBS, it is observed that JIVE SBS is one of the strongest social collaborative software available in the market with the unique features. And it has helped many big and small organizations by making them gaining a huge benefit and increased mass productivity by providing the social support [3].

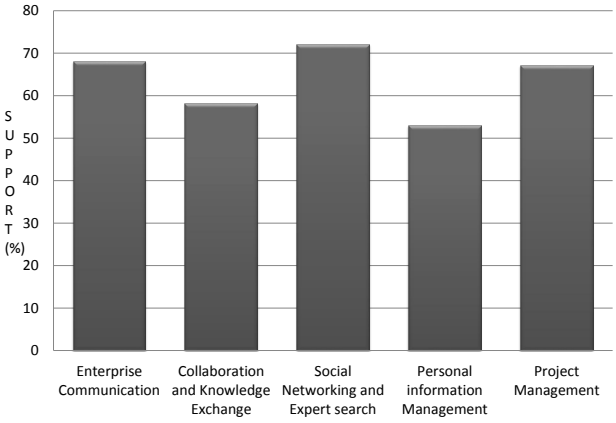


Figure 1. JIVE SBS 6.0 Functionality Support [4]

Figure: 1.0 shows the Functional evaluation of the Jive SBS 6.0. JIVE SBS provides effective communication tools. While, a support for knowledge base and knowledge is average because of the poor formalization of the discussion forum [3]. Besides, it provides very powerful tools for social networking and expert search which enables effective collaboration. It also seems that the provider has tried to armed the software with some good social network capabilities like personal information management and setting up your own look and feel of your profile, but it is not that effective as online social networks. It has a good support for the project management as it is easily extensible and integrable with the existing workspace.

Figure: 2.0 illustrates a technical support provided by JIVE SBS. From the figure it is clear, that the technical sup-

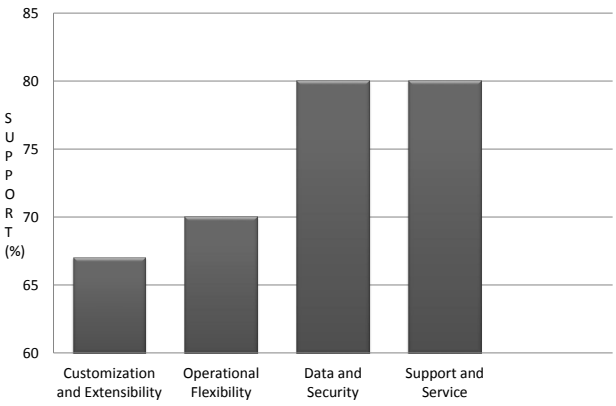


Figure 2. JIVE SBS 6.0 Technical Support [4]

port provided by Jive SBS is very strong. Customization ability of JIVE SBS is very good and it can be integrated with any software and support every kind of database. And it can be deployed as an internal software or can be used as a service. Moreover, 24x7 helpline shows their stronger customer support and service. Overall, JIVE SBS is an ideal collaborative solutions for the organization [4] .

2.2 Atlassian Confluence

Atlassian Confluence is also a social collaborative software like JIVE SBS, written mainly in java and developed and marketed by Atlassian [1]. Imagine a working environment, where you can send a less emails and dozens of meetings and still you can create a better result with your team, with confluence it is possible with glance of few clicks. It eliminates bottleneck, which flows your organization down and provide every team their own workspace to create and handle it. Share, discover and discuss content by bringing everyone to the one common platform is a special feature provided by confluence. Every member can have access to the project documents and meeting minute templates and editing rights are assigned and modified by the administrator. Moreover, it also enables separate wikis, pages for the different projects. The features like reporting dashboard, specs and mockups, event scheduling and attached documents review are also included in this software. Moreover, it is possible to make contribution for every team member with the use of powerful online editor. Simple design, simple features like drag and drop file sharing , copy and paste images directly from the internet, tag suggestions makes this software even more faster. Instant email notifications and custom activity stream for the team and projects are also a good feature, which shows their expertise in the area of social collaboration. To keep sensitive information private an administrator can decide exactly who can have access to and who can modify file and database [1]. Make public information open for all to see with unlimited number of anonymous users is the unique feature itself, which enhance it's ability to publishing customer facing documentation. Project management is one of the strongest features out of all the features provided by confluence. Team calendars for tracking the team leads, Projects and events, gliffy for the generation of reach diagrams makes project management easy and uncomplicated. Besides it provides a workspace to share your personal contents and tools to design your own look and feel. This is how it helps to breakdown the information and communication barrier exists between teams, departments and individuals by getting everyone in the same page [1]. There are also other solutions available in the market like Microsoft SharePoint, Google Apps, IBM connections and such. Functionality is consistent with all the software but their usage is depends on the particular organization in which they are deployed. Software can be deployed as internal server of the organization or a organization can request saas (software as a service), which almost all the software provider provides external hosting service. So choosing a effective collaborative software is a crucial issue of the current time and that extremely depends on the type and type of the organizations. Besides it also depends on the relation between all the users and the granularity of the usage of the software. Since most of the collaborative software are easily extensible and customizable they are suitable for almost all type of organization with some modifications.

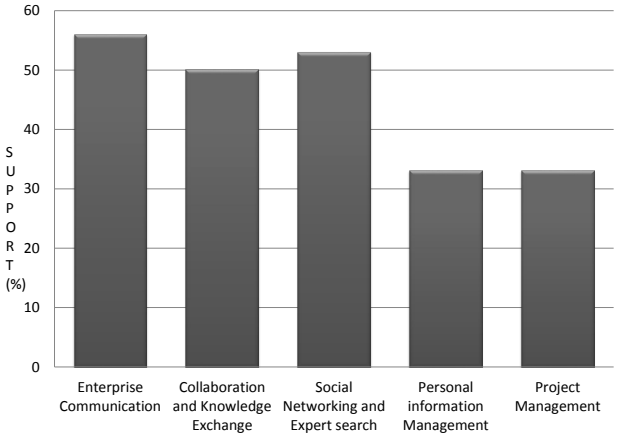


Figure 3. Atlassian Confluence 4.3 Functionality Support [4]

Figure 3.0 shows a functionality support in Atlassian Confluence. From the figure it can be seen that Atlassian Confluence has lower support to the functionality. confluence is a new collaborative software trying hard to catch up

with other software with the improvement in functionality. *Macros* [1] are the unique feature provided by Confluence. Macros are nothing but a piece of code. Using macros you can customize a tool support provided by confluence and make changes to them according to your need [1].

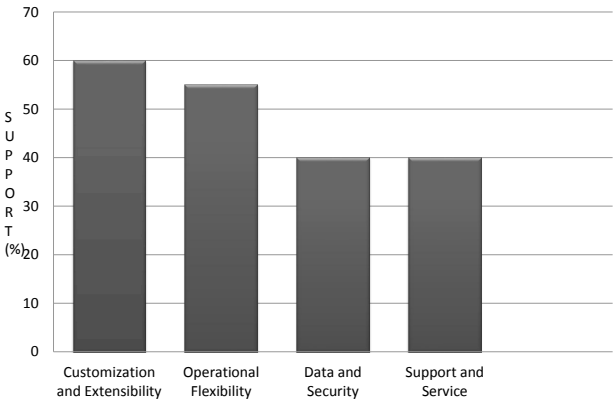


Figure 4. Atlassian Confluence 4.3 Technical Support [4]

Figure 4.0 describes a technical support provided by the Atlassian Confluence. In context of operational flexibility confluence has shown a noticeable improvement but it has a poor support for customization and extensibility as user has to write macros, which is very time consuming, moreover it requires technical knowledge to write the macros. Also, it has very limited support for the data and security. Customer support and service provided by Confluence are comparatively good for the small size companies [4].

2.3 Other Collaborative solutions

Big companies like Microsoft, IBM, Google have also jumped into the field of social software and they came up with good solutions like Microsoft offers collaborative services and system through a sharpoint while IBM connection is also a famous solution for the collaboration. They also provide the collaborative service(Software as service) by using their own cloud to store the data. Besides Google only provides the software as a service through Google Marketplace which is affordable for the small companies.

2.4 Effective Collaboration with e-Learning Platform

E-learning is a very broad aspect in the world of education. It share most of the concepts with online learning and distance learning though it is not same as both as it includes many other roles in it. There are many definitions available for the e-learning. According to Derek Stockley(2003) e-learning means *"The delivery of a learning, training or education program by electronic means. E-learning involves the use of a computer or electronic device (e.g. a mobile phone) in some way to provide training, educational or learning material"* [19]. Earlier LMS (Learning management system) were in use for the distance or online education but it was not as customized as it is supposed to be, because it was limited to particular courses or organizations [20]. E-learning is a solution to such problems which were raised in the past as it provides a open source for the E-learning platform. Because of enhanced information society and booming web technology e-learning has become a very broad standard which includes distance learning, online learning and a very modern blended learning. Blended learning can be argued as one of the main advantage of the e-learning as e-learning platform also capable to provide offline, face to face and individual learning. Christian Dalsgaard argued that social collaborative software tools can support a social constructivist approach to e-learning by providing students with personal tools and engaging them in social networks [8]. The idea of using collaborative social software with the e-learning platform is not very new as it was in the discussion since last decade [13]. Before selecting or choosing a collaborative system for e-learning one must consider some aspects of human nature and psychology because it can directly affect the usage of the e-learning platform. Also the size of the content and organization plays a vital role in the performance of the collaborative system. Moreover, collaborative software provides different tool supports like content sharing, discussion forum, expert help which can be used as a social and educational platform for the users or the student irrespective of their study field[8] and it also provides a huge support to group study and group work as we discussed in the above solutions. Wikis are the decent example which supports the influence of collaborative software into E-learning. The usage of the collaborative software is not just limited to the e-learning, but it also help to manage and administrate academia processes like admission, counseling, registration, conducting exams etc.

2.4.1 JISC digital media

JISC digital media is an organization which helps in higher education and further education sectors in United Kingdom by providing different services for e-learning. This organization was found by the JISC(*Joint Information System Committee*) which provides the services like training, webinars to the educational institutes [2]. E-learning mainly focuses on the normal users or the students by providing them tools to empower their ability and collaborative activities [8]. There are no specific tools developed for the educational purpose but tools like wikis, JIVE SBS, and Con-

fluence can be used as a collaborative platform with some immediate modifications.

3 Challenges faced by Social Collaborative Systems

As we know better solutions always come with big challenges, developers face many challenges during the development of collaborative system [22]. Design issue is one of the main challenges because it is the main user interface which can directly affect the success and production rate of particular organization. Bad tooling for communication or security elapses might result into a very serious loss for the organization. Also the performance of the collaborative software is also depends on the size of the organization, usage pattern and work style, and the size of content used by the organization. Performance can be also be varied by the size of network where collaboration software is installed whether on premises or existing cloud.

3.1 Design Process issues

Designing effective collaborative design tools is a challenging task. In addition to technical and implementation issues associated with concurrent and/or distributed work, designers are hampered by lack of data on how groups work together in software design. Collaborative applications are too often developed based on an individual experience of the designer, rather than on detailed study of the target user group and target tasks. This can result in tools that are neither useful nor usable. Even when user-centered design techniques are applied, the results are often tailored to the needs of single users, without sufficient support for collaborative work [10].

In order to support the collaborative work the collaborative software must support the number of different tools like communication tools, blogs, and forums. It should also support the different nature of the involved enities with the ease of transitional format [22]. The ability of supporting synchronous and asynchronous edit work style is the sign of a good design but is hard to implement. Besides the complex relationships between the users, are also one of the critical concern of the design phase. Developer must consider the following points before designing the collaborative software for any particular organization.

1. **Size of a Group:** It is very important aspect that the developers must take into consideration before designing the system. For larger group sizes, clustering becomes necessary. Some collaborative software does not support clustering, where as a software like Jive SBS supports clustering irrespective of a group size.

2. **Group location:** Location factor refers to the location of the people belonging to the same group. It is not necessary that all the people working on same goal, needs to be at the same place. They might be located to different location.

3. **Working style:** Working style describes how users work in the project and their way of integrating the work. It is hard to predict about the working style of the group but it is possible to characterize it.

4. **Timings:** Organizing and scheduling a work for the group is very difficult. so the collaboration software must provide am efficient tooling support in order to solve such problems.

5. **Group collaboration and communication:** In a software for large organization group collaboration feature is mandatory which enables users to easily get help from the external area and other groups, which also supports component based development.

3.2 Management issues

Team and work management using collaborative software is also a challenging task as it varies by different group size and type of the organization. To provide effective collaboration collaborative software must successfully handle the project and it should also provide the effective event management and event scheduling.

1. **Project Management issue:** Because of the cooperative working process and people with different specialization working on the same goal it is hard to manage the workspace for the project. For that each and every file and documents must have their information attached with them so anyone can access them. For example software designer and software developer in the same project accessing common document which edited or created by third party.

2. **Task scheduling and decision making:** Task scheduling is very crucial for teams working in different location of the same project. With presence of multiple experts in the project, decision process more complex as they may have conflicted views. Decision making tools are only important when people have unanimous opinions.

3.3 Authentication/Security Issue

Collaborative software provides a common platform for content and file sharing where people who are working on the same project can edit, update or review documents. To avoid loss and manipulation of the content, advance data security techniques are required. Moreover lots of security functions make software more complex.

3.4 Stability issues and workload due to complex relations between users

More often it is observed that the project does not contain the same type of people working on it. And relationship between such people becomes more and more complex that sometimes collaborative software fails to understand such relationships between the users and results into misplaced content and bad user co-ordination [6] . It is important for collaborative software to provide a tooling support to integrate people who are not part of the organization but working on the same project/goal. e.g Jive SBS provides such tools to include people who are not part of the same organization but working on the same goal.

Moreover frequent changes into customer requirements will have the indirect impact on the performance of the collaborative software as it needs to be customized with the change in requirements. Interoperability of the software is directly affected by such changes and it also results into a loss of time.

3.5 External dependencies

As collaborative software provides numerous functionality we can witness the growing complexity of such software. There can be many types of dependencies like social dependencies, technological dependencies and sometimes a user organization has to be dependent on the software and service provider. Such kinds of dependencies are always come into the picture after the deployment of the collaborative software in particular organization and results into a waste of time and money as well as decremented production ratio and lower leverage ratio. Investing a relationships between the actors and users has also become a very crucial issue since the trend of working is changing frequently and gathering more and more people to work on the same platform. It is very hard to characterize such issues [11] , Moreover, it also depends on the service provider when organizations use such software directly as a service. This approach is suitable for the small organizations who cant afford to have their personal network for collaborative software. There are number of things which need to be taken into account before using such software for particular organization. The first thing to be considered is the service and support provided by the social collaborative software providers. Product documentation and white papers are very helpful to solve such issues, moreover it should also consider the product maintenance and how frequently the product is being updated with the minimize cost. External networks and internal networks are also play very crucial role in the performance of the social collaborative software. Before designing and estimating the network size developers have to consider the following points.

1. **Security policy of the organization:** It is very important to manage the level of security and trust in the collaborative software where people share content and knowledge. Moreover a network should implement the needed security as per the interest of organization.

2. **Infrastructure of the organization:** Infrastructure of the organization includes almost all the points which were considered for the designing of the collaborative software.

3. **Work load distribution:** In order to balance the workload social software must provide the efficient social support and easily understandable interface to the users. It should also provide the different level within the same work space like centralized working streams and customized working streams. Centralized working streams provides the direct access to the workspace while customized working streams provides the local access to the workspace and can be easily managed by the project members.

4. **Size of data and information:** Last but not the least a size of data and content must be taken into an account to make software more effective and valuable. Because of the common workspace data is easily modifiable by any group member which leads towards the inconsistent and redundant database and results into a database manipulation. It is very hard to handle such database.

   While E-learning is also facing the same challenges as well as few more challenges as it is a growing technology in the current market. As the name itself suggests E-learning means an electronic learning with the use of hardware like computer, intranets and such. E-learning requires the technical knowledge of computer and information technology among the instructors and students. Due to the lack of such qualities it is not that much adopted by this time but it has produced very good results in the academia where it is adopted completely [5].

4 Conclusion and future work

4.1 Conclusion and how it can be relate to PG peer to peer

From the above discussion it can be observed that collaborative systems has made a very huge difference in the competitive market and changed the working trend by bringing more people on the same platform. It was possible due to the unique features like communication tools, social tools provided by the collaborative systems. Although there are some challenges like effective group collaboration, social support etc, which need to be investigated more deeply. So that we can make it more effective and organization friendly by adding the further tool support like group collaboration and project collaboration. While in e-learning, a limited knowledge of students and teachers about the use of

computer and collaborative system is a strong resistance in an adoption of e-learning in academia. But nevertheless, an adoption of e-learning has made a huge difference in academia and education administration by providing easy access to study material, exam handling and online consultation tools like JISC.

We can implement such collaborative system using the peer to peer concept where we can provide a unique experience to the normal users by providing more security and more user friendly interface as well as corporate users by making it more secure and providing a strong application support as the collaborative tools. Peer to peer also helps collaborative software to solve their security and privacy issue because data are being stored into the peer computers and also at the same time it provides a easy access to stored data.

4.2 Future Work

The paradigm of the collaborative software is indeed a very effective for the organization that uses the application oriented approach. But such social systems face some challenges due to the new disciples in the market, constantly changing human nature and working trend. It also need some investigation in the area of open source research which includes performance management, change in software characteristics and behavior due to the collaboration, and characterizing the work practice of the organizations.

References

[1] Atlassian Confluence. http://www.atlassian.com. Accessed: 2013-05-31.

[2] Jisc Softnet. http://www.jisc.ac.uk/fundingopportunities/opensourcepolicy.aspx. Accessed: 2013-05-31.

[3] JIVE SBS. http://www.jivesoftware.com. Accessed: 2013-05-31.

[4] Matrix For Social Software. http://www.socialsoftwarematrix.org/socialsoftwarematrix/. Accessed: 2013-05-31.

[5] P. Arabasz and M. B. Baker. Evolving Campus Support Models For E-learning Courses. *EDUCAUSE Center for Applied Research*, 2003.

[6] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. Characterizing User Behavior in Online Social Networks . In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009.

[7] L. M. Camarinha-Matos and H. Afsarmanesh. Collaborative Networks. In *Knowledge Enterprise: Intelligent Strategies in Product Design, Manufacturing, and Management*. Springer, 2006.

[8] C. Dalsgaard. Social software: E-learning Beyond Learning Management Systems. *European Journal of Open, Distance and E-Learning*, 2006.

[9] N. B. Ellison et al. Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication*, 2007.

[10] E. Francik, S. E. Rudman, D. Cooper, and S. Levine. Putting Innovation to Work: Adoption Strategies for Multimedia Communication Systems. *Communications of the ACM*, 1991.

[11] J. Grudin. Groupware and Social Dynamics: Eight Challenges For Developers . *Communications of the ACM*, 1994.

[12] S. Guha, K. Tang, and P. Francis. NOYB: Privacy in online social networks . In *Proceedings of the first workshop on Online social networks*. ACM, 2008.

[13] R. Klamma, M. A. Chatti, E. Duval, H. Hummel, E. T. Hvannberg, M. Kravcik, E. Law, A. Naeve, and P. Scott. Social Software for Life-long Learning. 2007.

[14] C. Lampe, N. Ellison, and C. Steinfield. A Face (book) in The Crowd: Social Searching vs. Social browsing . In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 167–170. ACM, 2006.

[15] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and Analysis of Online Social Networks . In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM, 2007.

[16] R. Pereira, M. C. C. Baranauskas, and S. Silva. A Discussion on Social Software: Concept, Building Blocks and Challenges. *International Journal for Infonomics*, 2010.

[17] M. Sánchez-Artigas, M. Arrufat, G. Parıs, P. G. López, and A. F. G. Skarmeta. SCOMET: A Middleware For Collaborative Working Environment in MANETS.

[18] C. Steinfield, J. M. DiMicco, N. B. Ellison, and C. Lampe. Bowling Online: Social Networking and Social Capital Within The Organization . In *Proceedings of the fourth international conference on Communities and technologies*. ACM, 2009.

[19] D. Stockley. E-learning Definition and Explanation, 2003.

[20] A. Tick. The Choice of ELearning or Blended Learning in Higher Education. In *Proceedings of the 4th Serbian–Hungarian Joint Symposium on Intelligent Systems*, 2006.

[21] S. Valenzuela, A. Arriagada, and A. Scherman. A Trend Study of Social Media and Protest Behavior: Facebook, Twitter and Youth Mobilization in Chile (2009-2012). 2013.

[22] J. Wu and T. N. Graham. The Software Design Board: A Tool Supporting Workstyle Transitions in Collaborative Software Design. In *Engineering Human Computer Interaction and Interactive Systems*. Springer, 2005.

Collaborative Document Editing in Peer-to-Peer Networks

Rapolu Manasa Veena
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email: mvrapolu@mail.upb.de

Abstract

Collaborative document editing has turned out to be an essential and crucial aspect in today's world where stakeholders work on same artifacts at the same time residing in different places across the globe. Existing asynchronous and synchronous collaborative editing tools like Wiki engines and Google Docs respectively are based on client server architecture that rely on a centralized server due to which several problems such as single point of failure, scalability, consistency, conflicts etc arise. Decentralized peer-to-peer systems has proven its ability to overcome these problems in many perspectives. In this paper, various approaches for the implementation of asynchronous and synchronous collaborative editing systems in peer-to-peer networks are discussed.

1 Introduction

As the need for collaborative editing is gaining its heights, often turning out to be indispensable in academia and corporate industries , the demand for collaborative editing tools in peer-to-peer network is increasing. Writing scientific papers , technical documents and planning presentations are few examples of collaborative activities which are the day-to-day routine in corportive world. Current wiki systems and version management systems such as CVS[1], Subversion[12] are based on the web architecture i.e, client-server model in which all the wiki pages and the resources that are published would reside on one single machine i.e, the central server machine.

This server-centric approach to enhance collaborative applications has several drawbacks such as *single point of failure* [8] where the whole system or network becomes inaccessible if the server is made unavailable. This may happen because of the system crash due to overload, power outage or an attack , as the server is centralized on which all the clients depend on . This system results in poor *scalability* as the server may run out of resources when unexpected number of clients use the service offered. This might lead to starvation of some users which are not served or might lead to failure of a system. Another major issue is the *high underlying hardware costs* to maintain the server, as the server is the one which keeps on running all the time. As the result of which it needs to be replaced when its service is ended or when it wears down. Also, the additional maintenance costs for the bandwidth, memory etc., for example if we consider any website which has gained much popularity then the issues regarding the bandwidth should be taken care of by ensuring that sufficient bandwidth is available to handle all the requests. Most popular sites like Wikipedia and Google spend several millions of dollars on bandwidth and hardware every year [11]. Server must also perform efficiently in order to overcome the high load imposed by various clients.Finally, the problem of centralized ownership is an issue which means that one single organization that owns the server hardware has full control of the network. The access privileges may be granted to other users in the network but can be revoked anytime by the owner. Also, limitation of the server to a fixed location ensures *poor performance* of the service offered when the connectivity to that particular location is bad or slow.

Due to the several drawbacks caused by traditional centralized systems, the necessity for the decentralized peer-to-peer systems seems obvious. In decentralized peer-to-peer systems there is no central server, thus overcoming the drawbacks caused in traditional client-server approach to the maximum. In peer-to-peer architecture, every peer or the system would act as both the client and the server. As there is no central control in peer-to-peer network, in which data is randomly made unavailable among multiple machines by maintaining replicant copies, changes cannot be made to the data unknowingly as the replicated copies exist on all the machines. Moreover, there is no problem of central ownership like in traditional approach since the data is made redundant on all the systems which leads to collective ownership instead. Maintenance costs are greatly

reduced here because there is no need of maintaining the hardware of the server. Bandwidth is also not an issue in peer-to-peer networks as it is disseminated among various peers present in the network. Hence peer-to-peer paradigm is well suited for the implementation of popular collaborative editing systems. Furthermore, the fully decentralized collaborative communication has become extremely popular now a days which has proven its benefits in various applications apart from file sharing, such as voice-over-IP, video streaming, and content distribution [8].

This paper is categorized into four sections where section 2 discusses about the version control management systems, in which centralized version control systems, distributed version control systems and peer-to-peer version control systems are discussed. Section 3 describes the approaches namely, Wooki [17] and DistriWiki [7] which are the wiki systems deployed over peer-to-peer networks that overcome the drawbacks of traditional centralized systems. Section 4 discusses about the Logoot [18] approach in detail and the fifth section follows the conclusion.

2 Version Control Management

The communication paradigm which is decentralized, i.e which takes place directly among the users, is nothing but the *peer-to-peer* communication. Contrary to the communication in client-server model, all the users, called as peers, act as both clients and servers at the same time, sharing the resources directly without any intermediate server [8].In collaborative communication, several users work on the same documents or projects concurrently at the same time leading to different versions of the artifacts which must then be made consistent in order to avoid conflicts among different users.
Version Control Management is the one which allows to track the evolutionary changes to the files, allowing the users to correct their mistakes or make modifications. This is an essential aspect in collaborative communication environment. Any modifications made by the user to the files in his/her working copy, place where the users work on artifact, are tracked and recorded by executing the *commit* operation. Other developers can then retrieve these changes made at any time by executing the *update* operation.
In the process of making changes to the content of the artifacts, versions* are being developed which are modified copies of the artifacts. This is the means for creating a new copy of an artifact rather than overwriting it. Conflicts which occur due to the concurrent work performed on a file by different users at the same time, are also detected and resolved using Version Control System(VCS). VCS provides the feature of restoring the entire project or a single file to any earlier stage i.e to an earlier versions allowing the users to correct any mistakes.

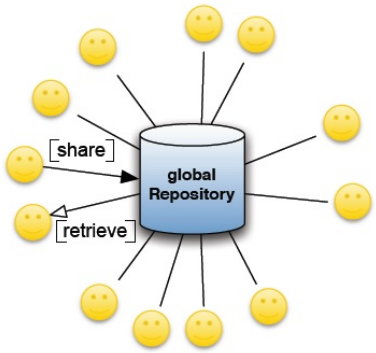


Figure 1. Basic principle of centralized version control systems[8]

2.1 Centralized Version Control Systems

Current version control systems such as CVS and SVN(Subversion) are based on centralized client-server comminucation and storage. In these systems, the central server stores all the data and responds to all the requests of the users which means that it stays online always. When the developer uploads data on the server, changes are detected, versions are marked and the conflicts are managed.The changes are then received by all other developers from the server. Figure 1 shows this.

The evident drawback with this type of communication would be the *single point of failure* presented by the server. Because all the data is reposited in this centralized server, failure to this machine would collapse the complete collaborative environment. Hence peer-to-peer communication paradigm became the most natural way of communication which is decentralized, with storage and communication among multiple users. Centralized version control systems are based on the client-server model where a central repository stores all the project(probably on a server) and all the developers *commit* their changes to this central repository. Committing the changes is to record or to save the changes in the global repository, where these changes are visible to other programmers. They can pull down the change as well, and the version control tool automatically updates the changes that were made to the contents of any files.

2.2 Distributed Version Control Systems

In order to overcome the drawbacks of centralized version control systems, Distributed version control systems(dVCS) evolved. The most popular dVCS include Git* and Mercurial*. In dVCS each and every individual user has a local repository on their own hard drive that contains the

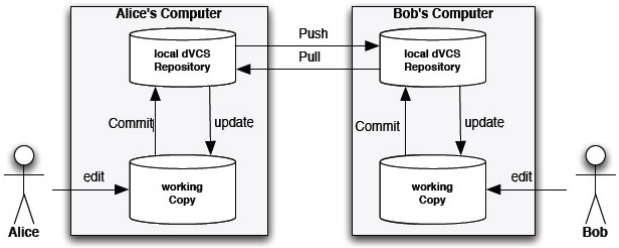


Figure 2. Sharing versions between participants of a dVCS [8]

full history of the project. All the versions of project's files are stored here. Artifacts are edited by the user in a *working copy*. A new *snapshot* is created in the local repository of the user when the commit operation is executed in order to store the changes made. Figure 2 [8] shows the sharing of snapshots of two users. The two users, Alice and Bob edit the project files in their own respective working copies and store the changes made in their local repositories by executing commit operation. Thus, their histories are created in their own repositories. In order to exchange their histories to each other *push* and *pull* operations are used. *Push* operation enables the snapshots of A to be sent to B whereas *pull* operation retrieves the snapshots from B to A if the operation is initiated at A, where A and B are two different users. (Snapshot: stores the meta data about commit i.e it stores the author's name, name of the committer and a comment[thesis])

Figure 2 shows the version sharing between two users, namely Alice and Bob who are editing the files in their working copies. By executing the commit operation, snapshots are generated which store the history of their changes made in their own respective local repositories. These histories are then exchanged to each other by using 'push' and 'pull' operations in between their local repositories. Each time the users commit their operations, latest snapshots would be generated in which updated artifacts will be stored.

The main drawback with the distributed version control systems is correctness [18] i.e a system is said to be correct if it converges eventually. Eventual consistency says that all the sites in the network contain identical data when the system is idle. DVCS does not obey this property (as all the users have their own local repositories and all of them may or may not have the latest versions of the files, which doesnot lead to correctness) As in the distributed version control systems every individual users have their own local repositories which are stored on their own system's hard disk then many problems may arise. For example, if any project has a very long history then downloading the entire history takes impractical amount of time and disk space. If the project contains many large, binary files that cannot be compressed easily, the space needed to store all versions of these files can accumulate quickly[internet]. Thus, storing all the versions of the files becomes a bottleneck.

2.3 Peer-to-Peer Version Control Systems

Peer-to-peer version control systems overcome the overhead of maintaining local repositories on every individual's system. This is done by combining the serverless operation of distributed version control systems with the *global repository* of centralized version control systems [8]. File sharing is well carried out in peer-to-peer networks like BitTorrent, eMule and many others. In this process of sharing files, a file becomes retrievable only when the users are online, offering this file. The peers who retrieve a file offer it to remaining other peers, being replicated in an unstructured way [8]. In the next section, we discuss about the fully decentralized peer-to-peer version control system, PlatinVC, in detail.

PlatinVC

PlatinVC [8] (refer to Figure 3) is a fully decentralized version control system that provides high consistency degree, efficiency and robustness. While developing a project, it is always necessary to keep track of the phases in the project i.e from the requirement phase to the implementation phase or testing phase. Artifacts in the project gets updated and new versions are also developed, as the project evolves in different stages. These changes are tracked by using the *traceability links* [9]. The platinvc is built based the decentralised and the centralised version control systems.

The user working on the project edit the working copy in his system and commit the changes,which is saved in the local module.When the user wants to share the changes, then he should push the content from local module to the peer-to-peer module which in turn is pushed into the virtual global repository.This virual global repository has all the modules of the project and it is not present in the single user machine. It is distributed among all the users using the PlatinVC.

The following are the design mechanism that are used by PlatinVC which makes it effective:

The *concurrency control* mechanism used by PlatinVC is optimistic concurrency control, in which multiple users can simultaneously modify the same artifact. If there is any conflict for the same artifact, then the user has to copy the artifact in specific version from the repository, modify it and merge it with the new changes that has been done by the user and then update it in the repository with new version.This approach is known as *copy-modify-merge*.

The repository distribution and partitioning used in PlatinVC is *Maintainer Based Approach*[8], which gives a solution to the concurrent commits by a central controller, known as *maintainer*. The maintainer decides which

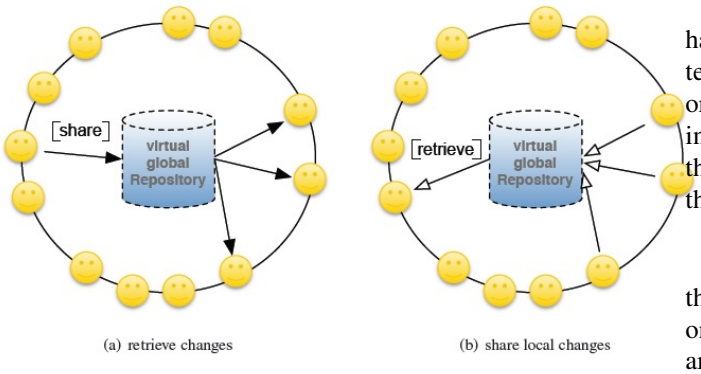


Figure 3. Sharing and retrieving versions using PlatinVC [9]

changes are committed, if there are two concurrent version conflicts for a single artifact. The artifacts used in the projects are stored as a folder. The maintainer holds the responsibilty of tracking the updated artifacts, which are located in the folders. If the peer executes *push* or *pull* operation globally, then it has to contact the maintainer.

As PlatinVC is built based on distributed version control system, it has almost all the features of it. Some of the additional features of PlatinVC are as follows:

Automatic Isolation of Concurrent Work:

The software project consists of many artifacts which are stored in many different folders in the repository. As there are many artifacts in the project, there arises many conflicts to manage all the artifacts concurrently. If only one developer is working on the project, then there is no chance of conflicts as the developer knows which artifact has to be changed. When two or more teams work on a project, then splitting their changes is more challenging. In centralized version control systems, the 'development lines' is split into branches. Once when the entire team working on the project completes, branches are then merged. Merging the branches in a very large project with many changes in artifacts is not easy. PlatinVC solves the problem by not creating the branch and it follows two workflows as *isolated folders* and *postponed branches* [8].

Working Offline:

Even without the internet connectivity, user can make changes and use PlatinVC to track the changes that are made. After making the changes to the artifact, the developer can push the content to the peer to peer module. As soon as the developer gets the internet connenctivity back, it joins the peer-to-peer overlay and share the changes globally with other peers.

Interoperability:

Like the distributed version control systems, PlatinVC has the ability to interoperate with other version control systems though exchange of their snapshots and synchronizing or in sequential batch process. After the local module gets interoperable with the external Version Control systems , the snapshots can be introduced globally and shared among the other participants.

Backing Up Redundant Artifacts:

In PlatinVC ,there is less chance of a copy to get lost, as the artifacts are stored in multiple users computer. Even if one of the user completely loses all the artifacts, the copies are stored in all other remaining peers. There are 'n' number of replicas for a single artifact. This redundant storage of artifacts reduces the chance of the artifacts to recede. However, only the recently committed snapshot might get lost.

Traceability Links: [9]

Many different phases exist in a project which should be linked to get the continuity of the all the stages in a project. PlatinVC uses these links that are created between any items like artifacts ,folders of different versions. PlatinVC enables to track the changes in the evolution of the project as well as the traceability links used.

The items like artifacts or folders are connected and it creates a companion file which has a generic name with following structure formed by "." prefix,followed by the item's name and with suffix "<-Linkfile". These companion file has the information regarding the links (information stored by link).A link document is created for each every link. If there is a link between A and B item,then a companion file with the name ".A<-linkfile" is created and has a link to the item B as "A<-link->B".The same holds for the item B also. Usually,these linked documents are stored with filename having suffix ".linkdocuments".

Whenever there is a change in an artifact, there is an update to a new version of the link document. These links always point to the recent version of the linked document. PlatinVC is robust and scalable regarding consistency,freshness and performance.

3 Asynchronous Collaborative Editing Systems in P2P Networks

Wikis permit asynchronous collaborative communication allowing everyday users to create and edit any page in a web site. They are the web pages that can be viewed and edited by any user having access to the internet with a web browser.

3.1 The Wooki

The currently existing collaborative editing systems in which wikis are the most popular, allows users for the easy creation and modification of the content on the web. This makes wiki an effective tool for collaborative writing.

Current wiki systems are centralized which causes several drawbacks which were seen in previous sections. Our goal is to replace the traditional architecture of wiki system by a peer-to-peer network which is decentralized, in which a centralized server is replaced by multiple systems called peers, which builds the whole wiki system robust and fault tolerant. In this system, the data is replicated on all the peers thus overcoming the overhead of heavy load on the centralized machine like in traditional approach. Furthermore, the underlying costs for the hardware can be shared among all the systems in the network. Peer-to-peer systems rely on replication of data among all the sites which ensures scalability and availability of the resources. But, because the data is replicated on all the sites in the network, that eventually leads to inconsistency when the data is modified by the multiple users concurrently. Hence ensuring the replicated consistency among the wiki sites would remain as a problem here, for which many optimistic replication [14] approaches are used for merging the concurrent updates safely. A system which is optimistically replicated must obey eventual consistency [14] i.e. all the sites contain identical data when the system is idle. This property is not ensured in distributed version control systems for which they are not used in building a peer-to-peer system. Hence woot [11] algorithm is developed for ensuring optimistic replicated consistency in a peer-to-peer environment. Here we discuss about the wooki, which is a functional wiki system deployed on peer-to-peer network, that is based on this approach.

Wooki is a wiki engine that is deployed over an *unstructured peer-to-peer overlay network*, whose network topology is shown in Figure 4(b). It follows the replicated repository approach in which all the wiki articles of all versions are stored by each peer. When there is any change made by the user to the content of an artifact then all the changes are propagated to all the peers in a line by line fashion. [8]. Wooki is dynamic in nature where any peer can come and leave the network any time. Every site has uniquely identified *siteid*. As many sites exist in the network, each site stores the same copy of data of wiki pages of other sites.

Disseminating the modified content of the artifacts to all the peers in the network ensures eventual consistency, but the flooding used in unstructured network doesnot guarantee that the modified content reaches to all the participants. The version control mechanism in Wooki handles only the updates made concurrently to the artifacts but the version history is not tracked. [8]

3.1.1 Wooto Algorithm

Wooto approach is an optimized version of woot algorithm in which each and every wooki page is assigned a unique identifier named *page id* when the page is created. If any site creates same pages with the same name as some other sites concurrently then their data will be most automatically by the wooto algorithm.

Every page in wooki contains a 4-tuples sequence, <*idl, content, degree, visibilty*> where each tuple represents each line in a wooki page.
idl represents unique identifier of a single line in a page, which is represented as a pair value with site id and logical clock [3] as parameters. This logical clock is assigned to each and every page in wooki which is incremented every time when an operation is generated.
content indicates the data present in the wiki line which represents string.
degree indicates an integer which is fixed when the line is generated.
visibilty indicates a boolean value which says if a line is visible or not as in the wooki approach, lines are not deleted, instead they are marked as invisible. So, if a line is invisible then the visibility value would be false, else true.
For example, if we consider a wiki page as,

- Computer Science
- University of Paderborn
- Masters/Bachelors

Assuming that the lines were generated on site number 1.

The page will be inventually stored in the system as:

- ((1,1), computer science, 0 , true)
- ((1,2), University of Paderborn,1,true)
- ((1,3), Masters/Bachelors,2,true)

Insert and *delete* are the two operations of a line that the wooki handles inorder to manage the storage model. 1. Insert (page id, line ,lp, ln) where page id is the respective page in which the line, <id1,content,degree,visibilty> is to be inserted. Lp and Ln are the line ids of the previous and the next lines respectively. 2. Delete(page id, idl) makes the respective line identified by idl in the respective page identified by page id as invisible. This operation sets the visibility to *false*. When we create a new page in wooto it initializes two special lines called as L_B and L_E. L_B is treated as a beginning of the page and L_E is treated as the ending of the page. Let *x* be the site and *P* be the page of a wooto, an insert operation needs to be performed on a page and is generated onsite x between the lines A and B where Line A and Line B are the two lines of an wooki page.

INSERT OPERATION = Insert (p, <(x,++ $clock_x$), Content, d, true> idl(LineA), idl(LineB)) [4]

Where d= max(degree(LineA), degree(Line B))+1
The Wooto algorithm ensures that the changes made to the artifacts are applied in the order in which they were made.

3.1.2 User's Operation

Users do not edit the model directly as in traditional wiki, where they make all the modifications they wanted and then save. They can view only the content of the visible lines when a page is opened for editing.

Diff algorithm [10] is used between the page requested by the user at edition time and the page that is saved by the user in order to detect the operations and then we translate these operations into Wooto operations. Translations of operations is done in such a way that deletion of line number n means to delete n th visible line and insertion of line at position n translated into insertion of the line between (n-1) th and the n th visible line.These translated operations are then integrated locally and broadcasted to other wooki sites [17]

3.1.3 Broadcast of Wooki

A broadcast protocol is used for the dissemination of operations as in [2]. Probabilistic broadcast algorithm is combined with an anti-entropy algorithm [4]. To manage membership and to quickly disseminates the updates on the peer-to-peer network we use probabilistic broadcast and to recover the missing updates of the sites that were either offline or crashed, we use anti-entropy algorithm. Broadcast protocol plays a vital role which must be scalable and reliable in order to be deployed on a peer-to-peer network. We must replace standard probabilistic broadcast of [2] with light weight probabilistic broadcast (lpbcast) [5] because in peer-to-peer network nodes may join and leave the network dynamically. This algorithm provides warranties that the sent operations will be reached to all the connected nodes in the network [17]

3.2 DistriWiki

DistriWiki[8] is a peer-to-peer wiki which is decentralized in nature, overcoming the limitations of the web as in client-server architecture. Present popular wiki engines like MediaWiki, TWiki, etc share the architecture of the web as they are implemented as an extension of the web server in which there exists a single server to which all the clients are connected in the network. Here all the wiki pages and the resources are reposited on one single machine i.e on the central server.
The main purpose of wikis are to provide write access to all the users who write, edit and share the documents in a site whereas the traditional wikis are based on the web which provide write access only to one organization i.e the owner of the site which does not fulfill the desired purpose. Hence peer-to-peer networks suits best for the implementation of the wiki engines in which every peer is treated as both the client and the server where all the users are granted permissions for the write access.
The requirements for the wiki network to be implemented on peer-to-peer networks are following: [7]

1.Redundant decentralization
2.Unique identification of documents
3.Efficient retrieval of documents
4.Usability
5.Compatibility with wiki concepts

If there is no central server in the system it is referred to as decentralized system. This concept is most important in peer-to-peer networking as the main motto is not to make the system unavailable in case of any failures by distributing the data among all the peers and making the data redundant in order to make the system reliable.

As the documents are concurrently modified by all the users, they must be uniquely identified by a global identifier which allows the system to notify the different versions made by the users and differentiate them, which would otherwise lead to conflicts.

Searching the documents and retrieving them efficiently is one of the requirements for the envisioned wiki network which strongly assists for the easy and reliable access of the network and none of the failed peers should affect the network. The traditional client-server system does not have efficient means for searching and retrieving the documents as there are no hosts with all the pages because of which the search function must span different hosts [7].

Since the users are accustomed to the existing wiki engines and the web browsers, the user interface should be made similar to the existing ones so that the users can easily switch, which is otherwise abstruse.

Traditional wiki concepts and semantics should be well preserved in peer-to-peer wiki systems.

DistriWiki is based on *hierarchical peer-to-peer overlay network*[14] JXTA. JXTA network topology is the hierarchical combination of subnetworks which are unstructured and the superpeer networks which are structured as shown in the figure 3 [8]. Subnetworks in JXTA are formed by all the systems that are in a local area network(LAN) whereas the systems which are connected over a wide area

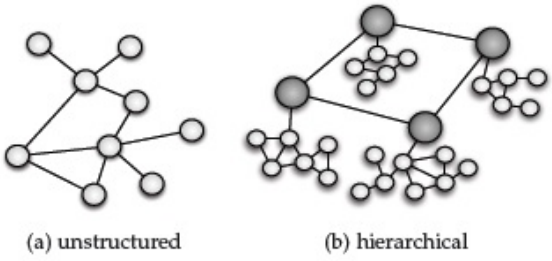


Figure 4. Unstructured and hierarchical peer-to-peer overlay network topologies [8]

network(WAN) form *superpeers*.

In JXTA, all the superpeers are identified uniquely with a number. Advertisements of the superpeers whose identifier values are close to their own identifier are stored by every superpeer, which have the information about the resources and the network services. DistriWiki uses these advertisements, which are the XML files in JXTA, in order to store the name and additional information of the wiki ariticles in it. These advertisements are identified by a unique name, type and an id by which desired resources can be searched for. JXTA offers an exciting feature, Resource Discovery, which is an easy and efficient way for searching the required resources on the network. All the peers have their own Discovery Service [7], for the discovery of resources, which is assigned when the peers join the network.

In DistriWiki, each and every superpeer knows all other superpeers whose identifier's numerical value is close to their's own identifier value. When an article is created newly, a JXTA advertisement is created along with a random number that is assigned to that article. This advertisement stores all the information pertaining to that article, including the peer which provided it. This advertisement is then sent to all the peers in the LAN network using the broadcast-mechanism of that underlying network. The superpeer to which all these peers in the LAN are connected and its neighbors having closest numerical value store the advertisement. When an article is being requested by a user, a broadcast-message is sent by the peer to all the peers in the network as a result of which all the published advertisements are received. Matching advertisement can be found easily from all the received advertisements because the advertisement stores all the information regarding the requested article.

Only the publisher who introduced the article can offer it to the users when requested, because of the reason that only the advertisements are replicated and not the articles. Even when the peer that is offering the article goes offline, the article becomes unavailable which reminds of the 'single point of failure' as in traditional server-based systems.(tell about the problems)

4 Synchronous Collaborative Editing Systems in P2P Networks

Online Collaborative document editing systems such as Google Docs and white boards facilitate real-time collaborative writing, enabling synchronous working with all the nodes in the network appearing simultaneously as how they type, i.e character by character. Real-time collaboration in Google Docs allows all the contributors to subsist in the same online space, working collaboratively towards a shared purpose. A newly added feature in Google Docs is that it shows a right side-panel indiacting the people who are currently editing the document, when more than one author is editing the document. Real-time collaborative editing [16] allows different users to edit and view the document from multiple sites at the same time that are connected by communication networks.

One of the major challenges in designing and implementing real time collaborative editing systems with a replicated architecture is consistency maintenance of replicated documents [16]. Consistency maintenance is thus one of the significant challenges in designing and implementing real time collaborative editing systems. Hence, a consistency model, which obeys CCI criteria, must be proposed as a framework for consistency maintenance in real-time collaborative editing systems. Hence, Logoot model [15] is introduced which ensures CCI consistency for linear structures.

4.1 The Logoot Approach

As the monolithic collaborative editing systems like Wikipedia are being sustained with a costly central servers, the aim for providing a decentralized peer-to-peer collaborative editing system came into picture. The currently exisiting approaches to build distributed collaborative editing systems does not provide scalability, for which we introduce Logoot approach. This approach renders scalabilty in terms of number of users as well as in number of edits. The CCI(Causality, Convergence, Intention) criteria [16] is being ensured in this approach i.e ensuring Causality, Consistency and Intention preservation.

Scalability is ensured in Peer-to-peer systems by relying on replication of data. Objects are replicated in structured networks such as Distributed Hash Tables or in unstructured P2P networks, a limited number of times or unlimited number of times respectively. In any case, whenever replication is done, it needs to be defined and copies must be maintained consistently. In this section, we lay out Logoot approach which is a new optimistic replication algorithm that ensures CCI consistency for linear structures, that tolerate unbounded number of replicas and that does not necessitate

the use of tombstones; deleted objects are kept as tombstones rather than deleting them from the document.

A collaborative editing system is considered as correct only if it obeys the CCI criteria. CCI criteria is briefly described here as follows,

Causality: This property guarantees that all the operations will be executed on every copy in the same order that are ordered by a precedence relation, as mentioned in the Lamport's 'happened-before' [6] relation

Convergence: Also know as eventual consistency. This ensures that the system meets the property which guarantees that all the replicated copies are identical when the system is idle.

Intention Preservation: If an operation takes place on any one replica then the effect of that operation must be observed on all replicas

In order to achieve commutativity, a solution that is based on total order between elements in the document is being proposed by the authors. To note, there exists two kinds of modification [15]
- *insert(pid, text)* inserts the line at the position identified by position identifier 'pid' with 'text' content.
-*delete(pid)* removes the particular line from the position identified by 'pid'.

Before, in order to maintain the total order between positions identifier, a tree structure is used. However, in order to ensure safe removing of elemens from this tree, tombstones were needed. In [13], authors referred to CRDT(Commutative Replicated Data Type) framework for which vector clocks were used to obtain this order.

The idea here is to assign position identifiers for each and every line with which a line can be deleted from the document without touching the order of the remianing lines.

Logoot Model:[15]

A Logoot document model encloses several lines which are defined by <pid, content> where pid indicates a unique position identifier and content indicates the text in that line. The beginning and ending of the document is represented by two virtual lines named $L_b and L_e respectively$.

The main idea of this approach is to generate a new position A by inserting a line such that P<A<N where P represents the position of the previous line and N represents the position of the next line. The position identifiers must be unique in order to allow the operations to commute. For any P and N, a position A must be able to be generated such that P<A<N, as there is a need for the user to insert a new line always.

Integrating Remote Modifications:[15]

The line can be safely removed from the document model by the integeration of a delete operation as the total ordering between remaining lines is not affected. Moreover, the removal of the line frees a position identifier which can be reused. The growing rate of the position identifier is reduced by this mechanism.

P2P constraints:[15]

Logoot approach requires to fulfill few constraints in order to be deployed on a P2P network. As the arrival and deaparture of the peers are frequent, the logoot approach should satisfy some constraints to support it. To obtain a casual dissemination mechanism, a lightweight probabilistic broadcast can be used. It supports these peer churns as it does not need any group membership mechanism and does not want to know the number of peers.

Logoot approach ensures real time collaborative editing over structured or unstructured peer-to-peer networks in an effective manner without the overhead of space and the need of tombstones. This enables the developers to utilize the space for useful purposes and relieves them from the overhead of maintaining garbage collector.

Asynchronous collaborative editing tools like Wikipedia have some control on concurrent modifications i.e few pages are protected from writing in order to reduce the malicious acts. However, in peer-to-peer environment, preventing the users from making concurrent modifications does not serve the purpose of real time collaborative editing. Hence, all the concurrent modifications made by the users are merged automatically which evidently leads to inconsistency. This certainly needs some user intervention to correct it, as the result of which number of edits are most important in peer-to-peer wiki than in the traditional wiki [15]

5 Conclusion

In this paper, we have discussed about the several peer-to-peer version control systems and also focused about the centralized and distributed version control systems disadvantages. As the peer-to-peer technology is becoming more popular with many decentralised features, it is very useful for the collaborative editing in real time. The peer-to-peer systems has many security concerns, as the data resides in all the peers in the network, inconsistencies arise when several users work concurrently on the same document.To overcome this problem several approaches have been introduced which were discussed in the above sections. As colloborative editing is very important for the development of projects in the large scale, efficient tools which overcome the drawbacks of the introduced approaches could be developed.

References

[1] CVS-Concurrent Versions System. http://www.nongnu.org/cvs/.

[2] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Trans. Comput. Syst.*, 17(2), May 1999.

[3] D. Cousineau, D. Doligez, L. Lamport, S. Merz, D. Ricketts, and H. Vanzetto. Tla + proofs. In *FM*, 2012.

[4] A. Demers, D. Greene, C. Houser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. *SIGOPS Oper. Syst. Rev.*, 1988.

[5] P. T. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight probabilistic broadcast. *ACM Trans. Comput. Syst.*, 2003.

[6] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7), July 1978.

[7] J. C. Morris. Distriwiki:: a distributed peer-to-peer wiki network. In *Proceedings of the 2007 international symposium on Wikis*, WikiSym '07, 2007.

[8] P. Mukherjee. *A fully Decentralized, Peer-to-Peer Based Version Control System*. PhD thesis, 2011.

[9] P. Mukherjee, K. Saller, A. Kovacevic, K. Graffi, A. Schürr, and R. Steinmetz. Traceability Link Evolution with Version Control. In *Software Engineering (Workshops)*, 2011.

[10] E. W. Myers. An O(ND) Difference Algorithm and Its Variations. *Algorithmica*, 1986.

[11] G. Oster, R. Mondéjar, P. Molli, and S. Dumitriu. Building a collaborative peer-to-peer wiki system on a structured overlay. *Computer Networks*, 54(12), 2010.

[12] M. Pilato. *Version Control With Subversion*. 2004.

[13] H.-G. Roh, M. Jeon, J.-S. Kim, and J. Lee. Replicated abstract data types: Building blocks for collaborative applications. *J. Parallel Distrib. Comput.*, 71(3), Mar. 2011.

[14] Y. Saito and M. Shapiro. Optimistic Replication. Rapport technique, 2003. Refer to rep:syn:1500 instead.

[15] t. . L. Stéphane Weiss and Pascal Urso and Pascal Molli. In *ICDCS*, 2009.

[16] C. Sun. Achieving Convergence, Causality-preservation, and Intention-preservation in Real-time Cooperative Editing Systems. *ACM Transactions on Computer-Human Interaction*, 1998.

[17] S. Weiss, P. Urso, and P. Molli. Wooki: A P2P Wiki-Based Collaborative Writing Tool. In *Proceedings of the International Conference on Web Information Systems Engineering – WISE 2007*, 2007.

[18] S. Weiss, P. Urso, and P. Molli. Wooki: a p2p wiki-based collaborative writing tool. In *Proceedings of the 8th international conference on Web information systems engineering*, WISE'07, 2007.

NAT-traversal Issues and Solutions

Weifeng Peng
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email: pengweifeng21@gmail.com

Abstract

Network address translation modifies IP address information in IPv4 headers and changes internal IP address into public IP address. NAT not only overcomes the shortage of IPv4 addresses in the Internet, but also protects internal devices from external attacks. However, external nodes cannot contact the internal nodes behind a NAT actively, which is obviously not acceptable in P2P system. In this paper, I present some feasible approaches (such as STUN, Relaying, UPD and TCP Hole Punching etc.) to solving the NAT traversal issue in our P2P framework.

1 Introduction

Today the client-server model is still the most popular network architecture and used by many famous companies in IT field, such as Google and Facebook. However, the client-server architecture excessively depends on the server, which makes the network not able to take full advantage of resources (processing power and bandwidth of clients) and have to face with the possibility of single point failure of the server. Consequently, the Peer-to-Peer (P2P) technology is getting increasingly accepted and becoming one of the hottest issues in recent years. To date, there are many well-known P2P softwares already, like BitTorrent, eMule and PPLive, which have millions of users all over the world and have been changing the habit of these users using the Internet.

Simply, in P2P network, nodes connect and communicate with each other without any server. However, Network Address Translation (NAT) makes it much more difficult. In order to deal with shortage of IPv4 address as well as secure internal nodes, NAT changes internal private IP address into public IP address. Therefore, external nodes cannot connect to the internal nodes behind a NAT directly, which is obviously against the main idea of P2P technology. Considering about 70% of users accessing P2P networks are behind a NAT system [1], it is quite necessary to tackle the NAT issue properly in P2P networks.

In next section, I will introduce what NAT concretely does and the types of NAT. Then, in Section 3, several NAT traversal solutions will be presented. At first, I will explain four basic NAT traversal techniques in detail (Connection Reversal, Relaying, STUN, UDP and TCP Hole Punching). Then, some more advanced and complex approaches, such as NatTrav and ICE, will be introduced. At the same time, the pros and cons of these approaches will be discussed. In Section 4, I will conclude all these solutions and try to figure out which NAT traversal solutions can be applied to the P2P framework of our project group.

2 NAT

The need for Network Address Translation (NAT) arises when a network's internal IP addresses cannot be used outside the network either for privacy reasons or because they are invalid for use outside the network [13]. Generally, NAT modifies the original IP address in IPv4 header into another IP address when a packet is going through a NAT device. And NAT keeps the address mappings in a table so that it can forward later received responses correctly based on the mapping table. Theoretically, sessions across a NAT are uni-directional, which means that only outbound sessions are allowed. Nevertheless, most NAT devices today permit network administrator to set static mappings in order to enable some pre-selected computers outside to initiate sessions passing through the NAT.

Additionally, NAT allows numbers of devices in a private network to access the Internet with one public IP address, which effectively alleviates the consequences of IPv4 address exhaustion before IPv6 is widely used, and applies to the vast majority of Small Office and Home Office (SOHO) networks and many other networks in our real life. For instance, most people's laptop computer and mobile phone access the Internet via their own router, free high-speed Internet in every room of a hotel, or WIFI in a college library.

According to RFC 3489 [12], there are four different types of NAT implementation:

- Full Cone NAT

Full-Cone NAT (also known as One-to-One NAT), means that once an internal IP address and port A:X, is mapped to an external IP address and port N:Y, all packets from A:X will be sent through N:Y. And any external host can contact with A:X by sending a packet to N:Y no matter whether it has previously received a packet from A:X or not.

- Restricted Cone NAT

A (Address) Restricted Cone NAT is one where all requests from the same internal IP address and port A:X are mapped to the same external IP address and port N:Y. Unlike a Full Cone NAT, an external host B:Z can send a packet to A:X through N:Y only if A:X has sent a packet to B before. For example, A:X sends a packet to N:Y on to B:Z. Then B:J can send a packet to A:X through N:Y. However other external hosts are not able to communicate with A:X.

- Port Restricted Cone NAT

A Port Restricted Cone NAT is similar to Restricted Cone NAT, all requests from the same internal IP address and port A:X are mapped to the same external IP address and port N:Y. But the restriction includes port numbers so that an external host B:Z can send a packet to A:X through N:Y only if A:X has previously sent a packet to B:Z. That is to say, if B:J has not received a packet from A:X before, B:J cannot connect to A:X even if A:X has sent a packet to B:Z.

- Symmetric NAT

A Symmetric NAT refers to one in which all requests from the same internal IP address and port A:X, to a specific destination IP address and port B:Z, are mapped to the same external IP address and port. Once the destination changes, the original internal IP address and port A:X sends a packet to the new destination C:W through a new external IP address and port P:M. Furthermore, only the external host which has received a packet from the internal host can send a packet back.

The first three NAT are cone-mapping types which keep the mapping between internal IP address and port A:X and external IP address and port N:Y and forward all outgoing packets from A:X through N:Y. Mappings are only removed from NAT mapping tables after a period of inactivity on N:Y [2]. Although Symmetric NAT is more secure than cone-mapping NAT types, there are still some studies believing that the industry is abandoning this NAT implementation type because it does not support Internet phones well and rarely applied to NAT devices in retail market place [2].

3 NAT Traversal

In this section, I address several basic NAT traversal techniques (Connection Reversal, Relaying, STUN, UDP and TCP Hole Punching) and some more complex and trendy approaches as well in order to alleviate the damage in P2P networks caused by NAT. At the same time, the merits and defects of these approaches will be considered.

3.1 Basic NAT Traversal Techniques

3.1.1 Connection Reversal

Connection Reversal is a very straightforward but limited technique [4], which works only if one of the peers is behind a NAT. For example, client A is behind a NAT and client B has a global IP address, and both of them can connect to the rendezvous server S, as shown in Figure 1:

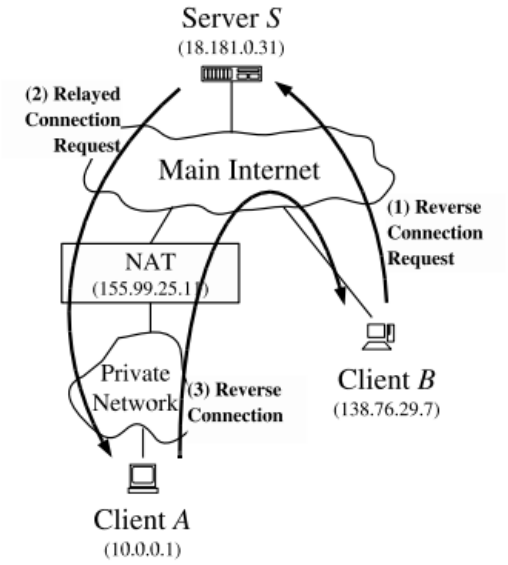


Figure 1. NAT Traversal by Connection Reversal [4]

Assume that client A has a private endpoint address 10.0.0.1:1234. Client A connects with a well-known server S at public IP address 18.181.0.31 and port 1235 via a NAT which allocates client A with a public endpoint 155.99.25.11:62000 for A's session with S. On the other side, client B is not behind a NAT. Client B has its own public IP address 138.76.29.7 and uses port 1236.

For client A, it is easy to establish a communication session with client B since B is not behind a NAT. A just needs to send a connection request to the NAT, then the NAT modifies A's IP address into 155.99.25.11 and forwards the connection request directly to 138.76.29.7:1236 through port 62000.

However, client B, in most cases, cannot connect to client A without the server S. There are two scenarios here: i) B might attempt to contact with the private endpoint of A, namely 10.0.0.1:1234 directly. But it will not succeed because the IP address is not a valid public IP address and will be dropped by the Internet. ii) B might send a connection request to the public endpoint of A, namely 155.99.25.11:62000. Nevertheless, A's NAT will block the request of B since only outbound connection is allowed to pass, unless the NAT is a Full-Cone type NAT or B is preselected by static mappings in the NAT table.

If client B wants to communicate with client A successfully, B can firstly send a connection request to 18.181.0.31:1235, asking the server S to relay the connection request to client A because S is connected with A. When A receives the request from S, it initiates a "reverse" connection back to client B across the NAT. Then the connection between client A and B can be done since the connection is originating from the inside of the NAT device and client B is not behind a NAT [3].

Considering majority of peers in current P2P networks are behind NAT devices, the shortcoming of Connection Reversal is obvious that it only works if one of the clients is not NATed. Therefore, Connection Reversal is not a recommended solution for NAT traversal in P2P framework. However, the central idea of this approach, using a reliable third party as an intermediary to help peers to set up a direct connection between them, is fundamental to another technique, hole punching, which will be described later [4].

3.1.2 STUN

STUN was originally defined in RFC 3489 [12]. At that time, STUN was entitled as Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs) and regarded as a complete solution for NAT traversal. It was mainly used for a peer to discover whether it was behind a NAT, acquire it public endpoint address on the NAT device and detect the type of the NAT (if a NAT existed). However, people gradually realized that STUN was not a guaranteed NAT traversal technique since it did not work sometimes. Therefore, RFC 5389 [10] has obsoleted RFC 3489 and redefined STUN in 2008. Now, STUN stands for Session Traversal Utilities for NAT.

Here is the central idea of STUN: Say a STUN client is connected to a private network and behind one or more NAT devices. And a STUN server locates outside the NATs in public network. In the first place, the STUN client sends a STUN Binding request to the STUN server. When this Binding request goes through the intermediate NATs and finally arrives at the STUN server, the STUN server obtains the public endpoint address of the STUN client on its outermost NAT, namely the source IP address and port number of the Binding request packet the STUN server receives. This endpoint address is also known as reflexive transport address. Then the STUN server XOR maps the reflexive transport address into the payload of a Binding response and sends the Binding response to the reflexive transport address on the outermost NAT. In the end, the Binding response will be forwarded to the STUN client by the NATs and the STUN client will learn its reflexive transport address in public network.

Additionally, RFC 5389 has added TCP and TLS-over-TCP transport protocols into STUN. Usually, STUN messages are sent in UDP packets avoiding from extra network overhead. Nevertheless, STUN messages are transported in TCP packets while reliability is desired. Further, Transport Layer Security (TLS) is used in some security-sensitive situations.

As STUN cannot guarantee its functionalities when it works alone and it is still limited by Symmetric type NATs, STUN is recommended to be used as a tool for constituting other NAT traversal solutions, such as ICE which will be introduced in later part of this paper, instead of a standalone solution.

3.1.3 Relaying

In Relaying approach, the two communication-desired peers just need to select a same reliable server (peer) and simply send all the messages via the relay server to communicate with each other. Obviously, Relaying approach works in a typical client-server way instead of P2P way to solve the NAT traversal issue. As shown in Figure 2:

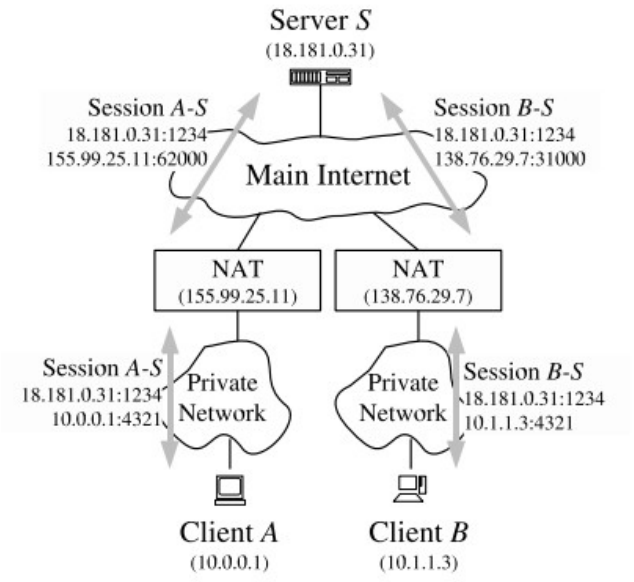


Figure 2. NAT Traversal by Relaying [4]

Suppose that client A and client B locate in two different private networks with private IP address 10.0.0.1 and 10.1.1.3 respectively. Client A is protected by a NAT at public IP address 155.99.25.11 and client B is behind another NAT at 138.76.29.7. Consequently, neither of these two clients can initiate a connection directly to the other one because both of them are behind NATs. Fortunately, either of them has set up a connection to a relay server S with IP address 18.181.0.31 and port number 1234. So, if client A attempts to send a message from its port number 4321 to client B, at first A sends the message to relay server S through the assigned public endpoint 155.99.25.11:62000. Then, S relays the message to B's public endpoint 138.76.29.7:31000 as an intermediary. Finally, the NAT of B forwards the message to B according to the established NAT mapping table in advance. If client B wants to communicate with A, just the same way around.

Apparently, Relaying approach is not a perfect solution for NAT traversal in P2P networks. Because this client-server-type technique has considerably high demands for processing power, network bandwidth, availability and security of the relay server to avoid from any bottleneck of the network or any single point failure caused by the server. Although Relaying technique would lead to excessive performance costs of the relay server and increasing latency of the communication between the clients, this Relaying solution is the most reliable method of P2P communication across NAT devices whether for TCP or UDP case, because it can work as long as both of the clients are able to connect to the relay server. So this inefficient but reliable solution can be applied as full-back strategy in our P2P framework in case that other approaches might fail in some complex and unexpected environment.

TURN [6], namely Traversal Using Relays around NAT, is a protocol that defines a method to implement Relaying in a relatively secure way. For security purpose, TURN does not allow users to run TURN server on well-known ports. In TURN, a TURN client uses a TURN server as a relay to send messages to peers and receive responses from peers. In general case, a client firstly creates and manipulates an allocation on the server in order to allocate the relayed transport address, namely an endpoint on the server, and peers can use the relayed transport address to relay messages to the client. After an allocation is created, the server receives TURN messages from the client, interprets the TURN messages into UDP datagram and sends them to peers. In the reverse direction, the server encapsulates UDP datagram from peers into TURN messages, then relays them to the client. For each client, there could be more than one allocation on the same server, which means multiple peers can send data to the client, through the relayed transport addresses of the client on the server. On the other hand, TURN is an extension to STUN protocol because TURN provides a method to traverse Symmetric NATs which STUN cannot handle.

3.1.4 UDP Hole Punching

UDP Hole Punching is a commonly utilized technique for traversing Restricted Cone and Port Restricted Cone NATs. Essentially, UDP Hole Punching uses a well-known rendezvous server as a third party to help set up a direct connection between two communication-desired peers which reside in private networks separately.

Before we go into the detailed UDP Hole Punching procedure, the concept of rendezvous server is addressed since it is a vital part in UDP Hole Punching.

Rendezvous Server: As mentioned in 3.1.1, rendezvous server plays an intermediary between two clients which are behind different NAT devices. Each of the clients has to set up an UDP session with a rendezvous server and maintain it before they create a direct UDP session with each other. When a client registers with the rendezvous server, the server obtains both the private endpoint and the public endpoint address of the client. The rendezvous server can learn the private IP address and port number of the client from the payload of the client's registration message and the public IP address and port number of the client from the IP and UDP headers of the registration message. Then the rendezvous server records these two endpoints. If the private endpoint of the client is identical with the public endpoint of it, it means that the client is not behind a NAT. When the rendezvous server receives a connection request between its registered clients later, the server will send the requested client's private and public endpoints to the requesting one as well as the private and public endpoints of the requesting client to the requested one.

Now, I would like to present the procedure of a general UDP Hole Punching case.

As shown in Figure 3, there are two client A and B locate behind two different NATs with private IP address 10.0.0.1 and 10.1.1.3 respectively. Both of them has connected the rendezvous server S at 18.181.0.31:1234 before hole punching. The client A's registration message is sent from 10.0.0.1:4321, then mapped to 155.99.25.11:62000 and forwarded to S by A's NAT. Similarly, the client B's registration message is send from 10.1.1.3:4321. then mapped to 138.76.29.7:31000 and forwarded to S by B's NAT. Consequently, these four endpoints are recorded at the rendezvous server S.

If client A wants to communicate with client B, A firstly sends a request to server S asking B's private and public endpoints. When S receives the request from A, S sends B's private endpoint 10.1.1.3:4321 and public enpoint 138.76.29.7:31000 back to client A. Also, S sends A's private endpoint 10.0.0.1:4321 and public endpoint 155.99.25.11:62000 to client B at the same time telling B

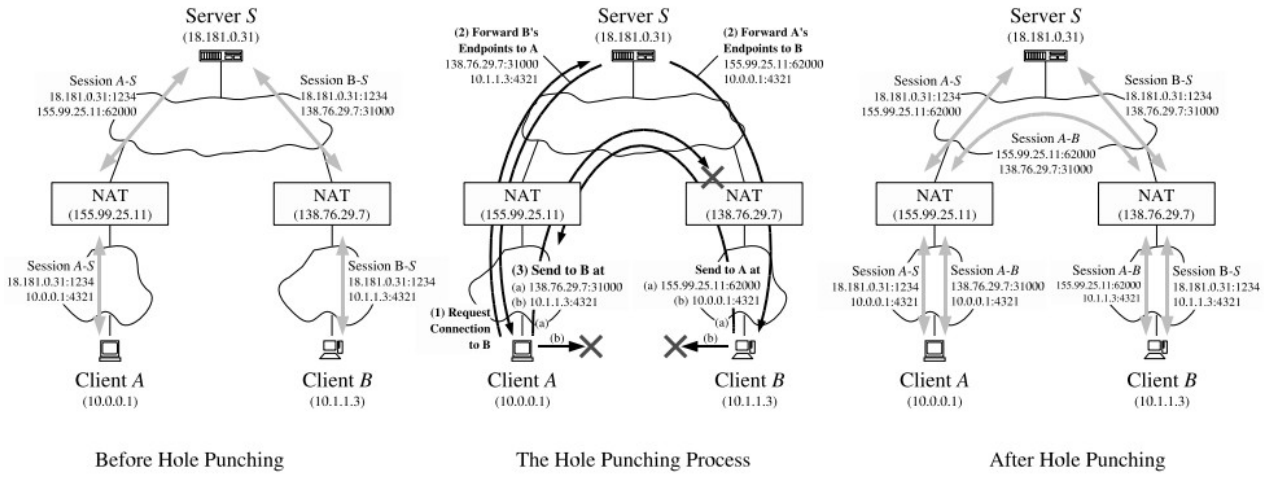


Figure 3. UDP Hole Punching [4]

that client A is requesting to contact with B. Therefore, each of the clients acquires the private and public endpoints of the other now. Then, A would send messages to 10.1.1.3:4321 and 138.76.29.7:31000 both. In the former case, the message could be received by another client which has same private endpoint with client B but in A's network. Obviously, the connection to the "fake" B is not what client A wants. So, it is necessary to involve an authentication mechanism to avoid from this kind of incorrect connections happening in practice. In the latter case, the message from A will get through A's NAT and be blocked by B's NAT as an unsolicited incoming traffic. Meanwhile, client A punches a hole in A's NAT for a new UDP session identified by the endpoints 10.0.0.1:4321 and 138.76.29.7:31000 on A's private network, and by the endpoints 155.99.25.11:62000 and 138.76.29.7:31000 on the public network [4]. Afterwards, client B would also send messages to A's private and public endpoints. The message towards A's public endpoint 155.99.25.11:62000 will punch a hole in B's NAT similar to the one in the A's NAT and finally be forwarded to correct destination by A's NAT since a hole already exists in A's NAT. Lastly, an UDP connection between client A and B is established and they can communicate with each other directly. This connection should be maintained by normal communications or periodic keepalive packets avoiding that the relevant translation state in the NATs is removed after a certain time (lifetime).

As we can see from the principle, UDP Hole punching does not support Symmetric NATs because client A needs to use the same public endpoint for both connections to S and B. And apparently it is unnecessary to utilize UDP Hole Punching in Full Cone NAT cases. So, UDP Hole Punching is a dedicated NAT traversal solution to Restricted Cone and Port Restricted Cone NATs.

3.1.5 TCP Hole Punching

TCP Hole Punching is described detailedly in [4]. The principle of using TCP Hole Punching solution is almost same with UDP Hole Punching introduced in last section, namely establishing and maintaining a P2P connection between clients which are behind different NATs with the help of a rendezvous server. Further, the connection established with TCP session is more robust than the UDP one because the lifetime of a particular TCP session can be precisely determined.

Here, the concept of TCP port reuse is explained before the behaviors of TCP Hole Punching since it is one of the most important part in the whole solution and makes TCP Hole Punching possible.

TCP Port Reuse: As the standard socket API is designed on the client-server model, there are some obstacles in front of TCP Hole Punching. First, a TCP socket is not allowed to initiate an outbound connection and listen for inbound connections concurrently. Also, a TCP port is usually available for a single TCP socket instead of multiple sockets. Fortunately, some special TCP socket options (SO_REUSEADDR and SO_REUSEPORT) provide a means to reuse TCP port. With port reuse, more than one TCP socket can bind to a same TCP port. As the example shown in Figure 4, each of client A and B has four TCP sockets binding to a local TCP port: a listen socket to listen for incoming connections, a stream socket connected to the rendezvous server S, two stream sockets connected to the other client's private and public endpoints.

Considering the procedure of establishing a direct TCP connection between two clients is similar to that of UDP Hole Punching (in Section 3.1.4), I prefer not to repeat it here any more.

Currently, there are just a few existing NAT devices sup-

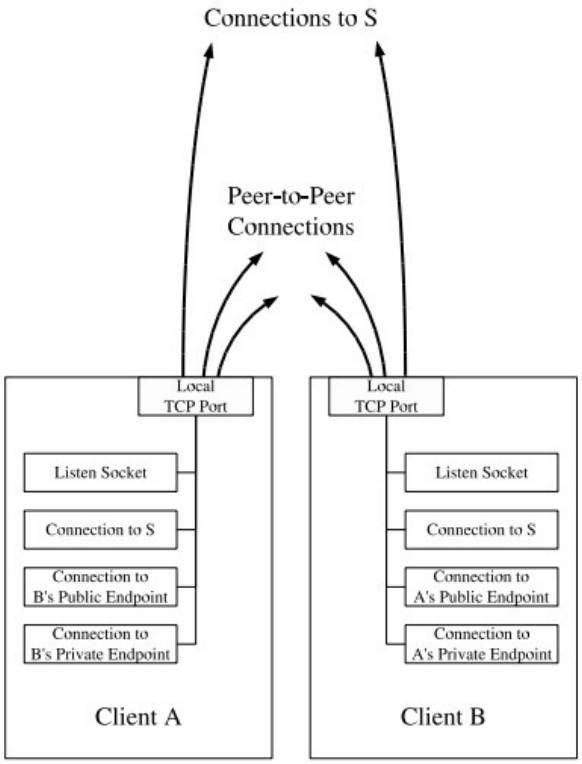


Figure 4. Port Reuse for TCP Hole Punching [4]

porting TCP Hole Punching [4] and it is not able to traverse Symmetric NATs same as UDP Hole Punching. TCP Hole Punching, in my view, may be used as an optional solution for our project in case that reliable connections between peers are desired.

3.2 Other Solutions

3.2.1 ICE

Interactive Connectivity Establishment (ICE) was proposed by Internet Engineering Task Force in RFC 5245 [8]. ICE is a compositive NAT traversal solution making use of STUN (in Section 3.1.2) and TURN (in Section 3.1.3) in order to work in any kind of network topologies. Generally, the two clients behind NATs can gather all their candidate transport (endpoint) addresses with assistant protocols (STUN and TURN). Then one client tries to communicate with all candidate transport addresses of the other client, using all candidate transport addresses of itself, so as to find out all feasible connection paths between the two clients.

Suppose that client A and B reside in different private networks, and both of them are behind NATs as usual. Furthermore, a TURN server S is utilized and it locates at the same public network host as the STUN server does. If client A attempts to communicate with B, ICE works as follow:

First, client A gathers its all candidate transport addresses. A can directly obtain its Host Candidate (private endpoint) from its local interface. And A sends an Allocate request to S. When an Allocate response is sent from the server S back to A, A acquires its Server Reflexive Candidate (public endpoint on A's NAT) and its Relayed Candidate (relayed endpoint towards B).

After client A has its all Candidates, it prioritizes the Candidates with certain algorithm and sends them to client B in a SIP [11] message. When client B receives the SIP message, B also gathers its all Candidates as A does and sends priority-ordered Candidates back to client A over SIP channel. Then, each of client A and B has enough information about both of their relevant Candidates.

Next, each of A and B pairs up its own Candidates and the other's Candidates into Candidate Pairs and checks these Candidate Pairs whether work or not based on a STUN request/response transaction. In this procedure, each client might get a new Candidate, namely Peer Reflexive Candidate, which makes traversal of Symmetric NATs possible. And after these checks, client A and B already can communicate with each other directly.

Afterwards, client A and B cooperatively prioritize the Candidate Pairs again with a specific mechanism in order to find out better connection paths.

ICE is a strong solution for NAT traversal since it combines functionalities of STUN and TURN as well as it supports both of UDP communications and TCP communications [9]. And ICE can work nearly in any network topologies and operates through any type of NATs. Although it might be more complicated to implement due to involving other more protocols, I consider that ICE could be one of the best solutions for NAT traversal when performance is prior.

3.2.2 NatTrav

NatTrav was raised in [2] and implemented as a JAVA package by the author. Basically, NatTrav aims at establishing direct TCP connections between peers, which are behind cone-type NATs, with assistance from third parties.

The architecture of NatTrav is shown in Figure 5. As we can see, there are three main roles in NatTrav, namely initiator, recipient and connection brokers. An initiator is a peer willing to set up a TCP connection to another peer. An recipient registers its address information along with an Universal Resource Indicator (URI), to be uniquely identified, on a certain connection broker. Connection brokers keep in touch with registered recipients and help them connect to a requesting initiator traversing NATs.

The procedure of NatTrav working is indicated in Figure 6. Assume the initiator I wants to set up a TCP connection

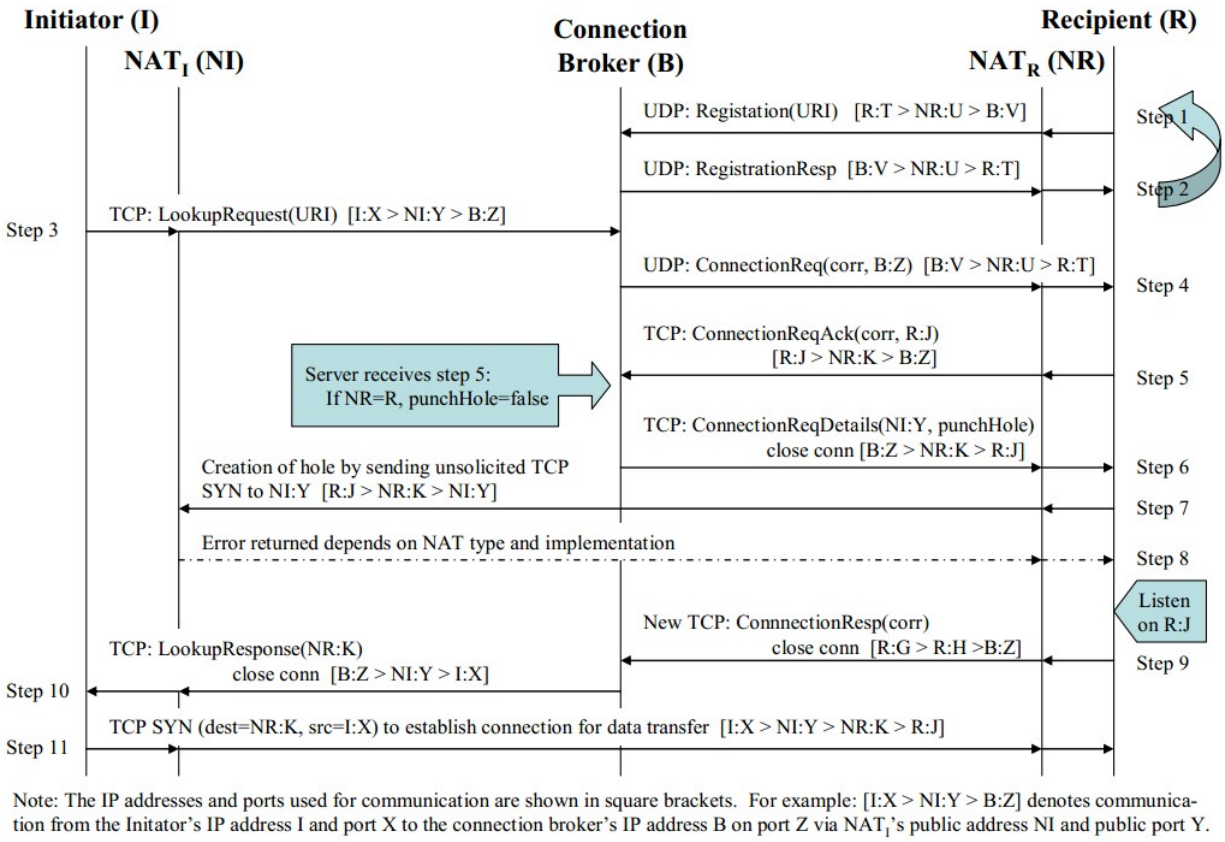


Figure 6. NatTrav Procedure [2]

to a recipient R identified by R's URI by asking a connection broker B. I and R reside in respective private networks behind their own NATs, NI and NR.

At first, recipient R needs to register its URI on the connection broker B before I initiates a lookup request, namely in Step 1 and 2. R creates an UDP (or TCP) connection to B and sends a registration message containing R's URI subsequently. When the connection broker B sends a registration response back to R successfully, the registration of R is accomplished.

In Step 3, the Initiator I connects to B using TCP and requests to communicate with recipient R identified by the URI. Actually, I can send the lookup request for R to any connection broker because those brokers that do not have the R's registration information will forward the lookup request to B. When B receives the lookup request from I, B also obtains I:X's public endpoint for TCP connections NI:Y, because NI is a cone-type NAT and NI always uses NI:Y for all TCP connections from I:X.

Then, in Step 4, B sends a connection request to R within R's registration UDP connection. The connection request includes not only an endpoint B:Z, which is used for following TCP connection to R, but also a correlator for tracking the initiator I's lookup request. In Step 5, R creates a TCP connection to B:Z and sends a connection request acknowledgment back to B, containing the same correlator and R's private endpoint for this TCP connection. When B receives the acknowledgment, B acquires R:J's public endpoint for TCP connections NR:K since NR is also a cone-type NAT. Now B can determine whether R is behind a NAT or not by comparing NR and R.

If NR equals to R, R is not behind a NAT (not shown in Figure 6) so that the initiator I can contact with R directly. B sends a message to R telling R it is not NATed and asking R to close the connection to B and set up a listen socket on R:J. On the other side, B skips to Step 10 directly and sends a lookup response to I containing the exact address and port of the requested URI (R:J). Afterwards, I can initiate a TCP connection to R successfully since B is not NATed.

If NR is different from R, R is behind a NAT, as Figure 6 shows. In this case, B sends the source IP and port number of connection request (NI:Y) to R and also tells R that hole punching is needed because R is NATed. Then B closes the connection to R in Step 6. When R receives the connection request details, in Step 7, R attempts to create a TCP connection to I's public endpoint NI:Y directly. Obviously, this

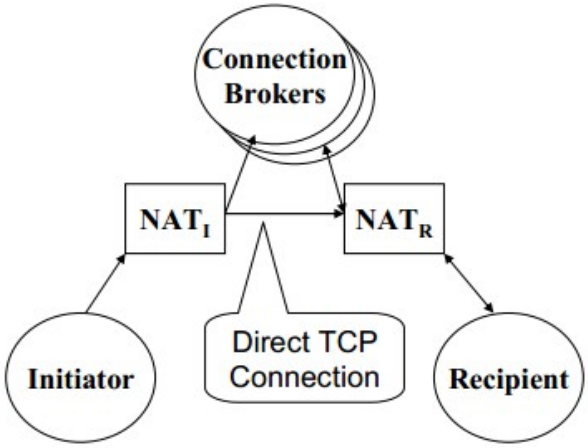


Figure 5. NatTrav Architecture [2]

attempt will be rejected as unsolicited traffic by NI. However, R punches a hole on NR in effect. After receives an error message from NI, R sets up a socket listening on R:J. Next, in Step 9, R uses another port to create a new TCP connection to B and send B a connection request identified by the correlator, then R closes this connection. In Step 10, the connection broker B tells I the public endpoint of R (NR:K) as a lookup response and closes the connection between B and I. Finally, the initiator I can establish a TCP connection to R via NATs.

This NatTrav solution mainly utilizes TCP Hole Punching to realize the TCP connections between NATed peers without any compromise on security, keeping infrastructure deployment and configuration, or using standard protocols and APIs. Besides, NatTrav can extend to support Symmetric NATs by using port prediction [5]. After all, NatTrav provides us a quite reliable option for NAT traversal.

3.2.3 Superpeers

This Superpeers approach was proposed in [14] in 2008. Essentially, it is based on the old version STUN protocol [12], which believes that STUN server is able to determine whether a STUN client is behind a NAT or not and detect the type of the NAT (if exists), for NAT traversal in P2P networks. Using this approach, each peer is supposed to learn the type of its NAT (or no NAT) and chooses the best NAT traversal technique according to both NATs' types when this peer attempts to establish a connection to another peer. So, there are two steps in this approach.

First, each peer requests a STUN server for its NAT information. There exist two kind of STUN server: public STUN server and internal STUN server. The former one refers to original STUN server located in public network. However, these public STUN servers might reside outside of the P2P network. Therefore, the approach creates some internal STUN servers dynamically into the P2P network to enhance the independence of the P2P network. Every internal STUN server consists of two so-called superpeers which are behind FC NATs or no NAT at all, and registers at the DHT of the P2P overlay network. In this Step, each peer can obtain the type of its NAT and its public endpoint on the NAT for usage in the next step.

Second, after every peer learns its NAT type, peers can select proper NAT traversal techniques relying on Table 1 and set up connections between them. The connection initiating peers' NAT types correspond to the leftmost column and the upmost row refers to the connection receiving peers' NAT types.

NAT	FC	RC	PRC	SYM
FC	Direct	Reversal	Reversal	Reversal
RC	Direct	HolePuching	HolePuching	HolePunching
PRC	Direct	HolePunching	HolePunching	Relaying
SYM	Direct	HolePunching	Relaying	Relaying

Table 1. NAT Traversal Dependent on NAT Types [14]

Theoretically, This approach is a relatively perfect solution for NAT traversal in P2P networks since it chooses the best NAT traversal technique for each specific situation dynamically. However, as I mentioned in Section 3.1.2, STUN is not used to detect NAT types any more because it has been found not a reliable way to do so. Therefore, this approach is not recommended to be used as a primary strategy in our project group before it proves the performance in practice.

3.2.4 UPnP

Universal Plug and Play (UPnP) [7] is a protocol based on many Internet standard protocols, such as TCP/IP, HTTP and XML so that it can work with any operating systems and programming language. UPnP can configure the NAT automatically when a new UPnP compatible device gets into a network spatially, assigning it with a private network address and showing it other devices' presence in the same network. Then the newly joined device is able to communicate with other active devices in this network. However, there is no authentication mechanism in UPnP by default, which makes UPnP extremely vulnerable. Consequently, many UPnP-support devices have turned UPnP off by default avoiding from potential security issues.

As can be seen, the characteristics of UPnP make it better to be applied to a trustworthy small home/office network instead of relatively complex P2P networks.

4 Conclusion

In this paper, several NAT traversal solutions are introduced for the purpose of facilitating connectivity between peers in P2P networks. The principle and characteristic of each solution are also included in order to enable our project group to select a proper solution in our P2P framework.

Connection Reversal (in Section 3.1.1) works only if one of the communication-desired peers is behind no NAT (or Full Cone NAT), which makes it improbable to be applied into practical networks that are much more complex and indeterminate. With the basic idea of Connection Reversal, UDP and TCP Hole Punching (in Section 3.1.4 and 3.1.5) build up a mechanism to realize UDP and TCP connections between peers which are behind well-behaved (Restricted Cone and Port Restricted Cone) NATs. NatTrav (in Section 3.2.2) mainly uses the functionality of TCP Hole Punching to develop a more powerful solution because NatTrav can traverse Symmetric NATs by using port prediction. On the other side, STUN (in Section 3.1.2) is used as a significant part in ICE (in Section 3.2.1) and Superpeers (in Section 3.2.3) approaches, mainly for obtaining the server reflexive endpoints of peers. ICE solution, which combines STUN and TURN, is capable of working in any network topologies and handling any type of NATs. And Superpeers, which depends on the functionality of STUN, provides an optimized solution at runtime. If all these solutions fail, Relaying (in Section 3.1.3) could be used. Although Relaying solution wastes a mass of processing power and bandwidth of a relay server, it can work as long as both peers are connectable to the server.

References

[1] M. Cadaco and M. Freedman. Illuminati-Opportunistic Network and Web Measurement, 2007.

[2] J. L. Eppinger. TCP Connections for P2P Apps: A Software Approach to Solving the NAT Problem. 2005.

[3] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-Peer (P2P) communication across middleboxes. 2003.

[4] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-peer communication across network address translators. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05. USENIX Association, 2005.

[5] P. Francis. Is the Internet Going NUTSS? *IEEE Internet Computing*, Nov. 2003.

[6] R. Mahy, P. Matthews, and J. Rosenberg. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN), 2010.

[7] B. Miller, T. Nixon, C. Tai, and M. Wood. Home networking with Universal Plug and Play. *Communications Magazine, IEEE*, 2001.

[8] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols, 2010.

[9] J. Rosenberg, B. B. Keranen, A. Lowekampand, and R. A. B. TCP Candidates with Interactive Connectivity Establishment (ICE), 2012.

[10] J. Rosenberg, R. Mahy, and D. Wing. Session Traversal Utilities for NAT (STUN), 2008.

[11] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol, 2002.

[12] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), 2003.

[13] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT), 2001.

[14] A. Wacker, G. Schiele, S. Holzapfel, and T. Weis. A NAT Traversal Mechanism for Peer-To-Peer Networks. In *Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing*, P2P '08. IEEE Computer Society, 2008.

OSGi and Dependencies: Loading and Using Software Bundles

Chander Prakash
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email: chanderp@mail.uni-paderborn.de

Abstract

This paper presents an overview of OSGi, discusses the architecture in detail, explains benefits of using OSGi from both developers and business point of view. It then presents some current OSGi framework implementations. This paper also explains what an OSGi bundles means and how to create that, what kind of metadata it contains. It also highlights some existing OSGi repository. It then presents some important aspects in Decentralization in Peer-to-Peer (p2p) networks which are Decentralized Dynamic Code Management for OSGi, Decentralized Service Discovery and a survey on resource discovery mechanism.

1 Introduction/ Motivation

Over the years, the programming has changed a lot. In early days, the programming languages were procedural, which contained a series of computational steps. Further, these languages were not able to hide their implementation details. A revolutionary step in the programming world was OOP[1]. The key element `class files`, see figure 1, were capable of hiding their implementation details and provided a better level of abstraction. A class contains methods, which can be public or non-public. All the methods are by default non-public. Only public methods are available to the outside world. All those languages which were capable of supporting OOP, allowed the programmers to show only those things which they want to be available to the outside world and hide other details.

Java is a very popular language which supports OOP concepts and the unit of modularity is JAR[2] file. By default these JAR files are not able to hide their implementation details as classes can do in Java. Let us assume that JAR files are also capable of hiding all the information as the classes can do. The possibility of JAR files creating a public contract in the same way as classes do and we can achieve all the benefits of information hiding at both during development and runtime, see figure 2. This is only possible when JAR files are running under a special platform/framework called OSGi[3] framework.

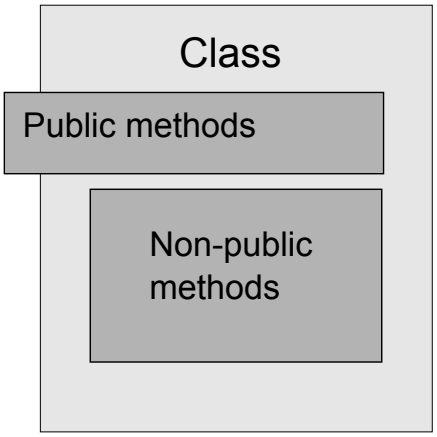


Figure 1: Class file in OOP language
Source: own figure based on [47]

The key part of the current PG is the ability to provide developer to develop Plugins and OSGi can be used to create Plugins for this PG framework hence OSGi framework must be understood. In the second section, we first look what OSGi framework is and what is the architecture of the OSGi framework. We then quickly look some of the benefits of using OSGi followed with some common myths and misconception about it. Then we will study some of the current OSGi framework implementations. Section three of this paper is dedicated to the OSGi bundles. We will see that what a bundle is, what kind of data it contains and how we can work with the information which a bundle contains. As decentralization is another key part of the current PG, section four deals with the decentralization in p2p systems. It tries to explain Decentralized Dynamic Code Management, provides a detailed overview on decentralized service discovery and finally it presents a survey on resource dis-

[1] Object Oriented Programming
[2] Java Archive contains Java class files, associated metadata and other resources
[3] Open Service Gateway initiative is a framework for Java

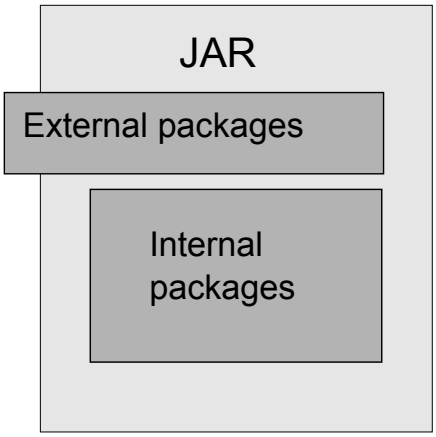


Figure 2: JAR file under OSGi
Source: own figure based on [47]

covery mechanism. Section five summarizes the paper and presents the conclusion.

2 OSGi Service Platform

The OSGi stands for "Open Service Gateway initiative" [58], which is designed for Java programming language. It is a modular system and it adds more flexibility to the Java programming language and gives it the functionality that does not exist in the standalone Java/ VM environments. This technology is so powerful that you can install, start, stop, update or even uninstall any application/ component without reboot and it gives the programmer even more flexibility that it can be done remotely. The applications/ components are in the form of bundles, which are the basic unit of modularity in this framework/ technology. The applications/ components are managed by API calls, which makes it even more flexible in terms of configuring components remotely. Using this framework, components can adopt accordingly to the updates that are made.
This dynamic component system [59] allows jar files to hide their implementation details from the outside world and at the same time it exposes its functionality to the outside world so that it can be used efficiently by others.
It is the first technology that successfully implemented the component system and solved many real problems related with software industries in a component based system. Due to the minimum overhead, OSGi technology helped in reducing complexity in the development process [59]. Under OSGi framework, developers has the freedom to wrap the details of JAR files and make them visible as per requirement otherwise they are hidden. OSGi also helped in easier code writing and testing. This technology also helped in reusing of functionality, simpler system designs, easy error detection and correction because it makes system so transparent that we can see what is running inside. The most important part is that it is been adopted by popular applications like Eclipse and Spring.
Initially the OSGi specifications had the focus area in service gateways but now it has been spread over a wide range of applications from mobile phones to the Eclipse IDE. There are so many other areas where OSGi is been used like automobiles, PDA's, application servers are some of the non ending list of OSGi service area.

2.1 OSGi Architecture

OSGi structure can be seen in figure 3. OSGi implementation directly the top of the JVM and provides mechanisms for service management, component definition, execution, management and life cycle control of modules. The most essential OSGi concepts are described below:

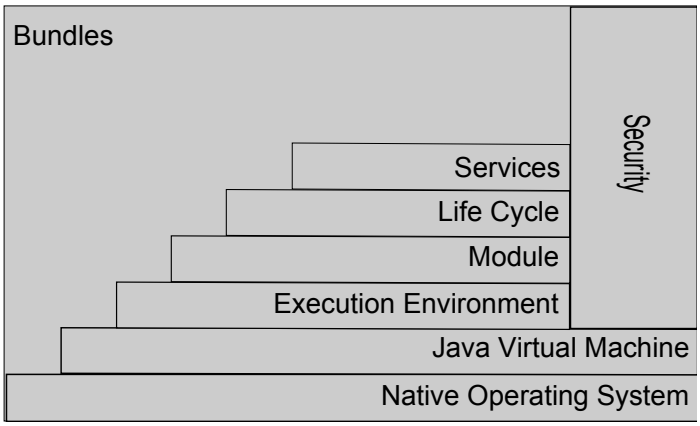


Figure 3: OSGi Architecture
Source: own figure based on [38]

- Bundles: Bundles are also JAR files but they contain some additional information like some OSGi specific `MANIFEST` header information and some OSGi classes that qualifies it as a bundle [39].
  A bundle contains around 20 header informations. It is not mandatory to provide all header information. The only mandatory header that must be provided is `Bundle-SymbolicName`. Section three provides detailed information on bundles.

- Services: The services layer connects bundles in a dynamic way by offering a publish-find-bind model for plain old Java objects [59]. A bundle can register its created objects under the OSGi enabled interfaces. A bundle can register for a service, it can get a service or it can listen for a particular service., see figure 4. In short, OSGi framework is a technique to access, register and manage the services.

  There is no restriction on the number of services a bundle can listen or register.

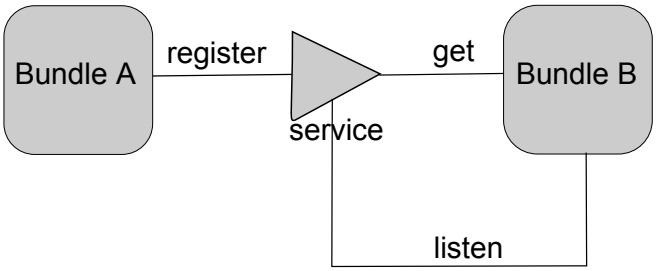


Figure 4: Service layer
Source: own figure based on [59]

- Life-Cycle: OSGi provides a mechanism to control the life cycle of the bundles through the API to install, start, stop, update, and uninstall bundles [39].
  The OSGi bundle life cycle has six different states. Please refer section three of this paper for complete description of bundle life cycle.

- Modules: This layer defines how a bundle can import and export code. Modularity makes everything local and prohibits the sharing of the things until and unless its not done explicitly. By default, JAR files does not hide any details from each other. It means all the details of a JAR file is available to the outside world. When OSGi is imposed on a JAR file, it hides it from other JAR files. But when we want to expose some functionality to some other files, it creates a contract and allows only that part to be accessed. Whenever a bundle wants to use some other files, it must import all those packages. By default there is no sharing.

- Security: The layer that handles the security aspects. This layer limits the bundle functionality to predefined capabilities [58]. It enables the full security in the OSGi framework. In OSGi, all the life-cycle operations are fully protected with the security architecture imposed by OSGi framework.

- Execution Environment: This layer tells the environment under which the bundle will run. Though OSGi is implemented for Java programming language, thus it tells under which Java environment the bundle is going to run. This may be, for example Java EE or Java SE. It also defines what methods and classes are available in a specific platform.

2.2 Benefits of using OSGi

OSGi is a technology which can work for so many environments and it is a well proven and absorbed by so many renowned names in the field of software development. With the countless applications under its name, there is no surprise that this technology has so many benefits. Here are some benefits of using OSGi from the list of countless benefits.

- Reduced complexity: In OSGi, the basic unit of modularity is bundle. And all bundles are modules, which are capable of hiding their implementations from other bundles. Hence the developers are more free to just use well defined bundles in their applications. It just make it that simple that the developers have to call the functionality of a bundle by specific interface.

- Reuse: The best part of the OSGi framework is the reuse. A developer can import as many bundles in the project, on the other hand, one bundle can be used in so many projects. There are lots of open-source projects which provide ready to use OSGi enabled libraries.

- Dynamic nature: The real world is totally dynamic in nature and so OSGi is. OSGi framework is highly dynamic. The bundles can be updated without the need of restarting, which is an important aspect and does not supported by simple Java/ VM. Thus OSGi really fit for the real world.

- Dynamic Updates: OSGi framework is highly dynamic in nature. Bundles can be installed, started, stopped, updated and uninstalled without requiring any reboot, which is the functionality provided by the OSGi framework. It makes OSGi suitable for so many critical applications.

- Versioning: All Java developers are aware of the common problem of JAR hell [57] which can be described by as the phenomena where class loading process fails due to versioning reasons. This problem was solved by the OSGi framework because under OSGi framework, the header named `Bundle-ManifestVersion` provides a proper versioning system. And while importing or exporting a bundle, one must specify the proper version.

2.3 Some myths and misconception about OSGi

We have so many technologies and no technology is perfect, OSGi too does not make any exception. And before adopting any new technology, peoples always try to find that if the technology they are going to use is simple or not, if there are resources available,does it has any standards and so many other things. And the main thing is that the peoples perceive it to be too difficult to understand or to implement without even knowing it. Here are some myths and misconceptions about OSGi which sticks with the people too often:

- OSGi is complex: From first impression OSGi seems to be too complex because you have to manage

`META-INF/MAINFEST.MF` file. This file contains entries that we generally don't have to worry about and for using OSGi, we only need to add one additional entry in this file for a JAR file to qualify as an OSGi bundle (Bundles are unit of modularization in OSGi): `Bundle-SymbolicName`.
`Bundle-SymbolicName` is used to uniquely identify a bundle. Though we have several headers that can be included in a bundle's `MAINFEST.MF`, only a few of them is being used. Also much of the management can be automated by using BND, a tool for managing bundles.

- OSGi is new: OSGi technology is almost 13 years old but never got too many attentions from Java developer's community. The OSGi Alliance was founded in year 1999 and after one year, in 2000 OSGi 1 was released. It is even older than the Microsoft .net technology, which was released in 2002. Here are history of OSGi framework development, see Table 1.

Year	Release
March 1999	The OSGi Alliance was founded
May 2000	OSGi Release 1
October 2001	OSGi Release 2
March 2003	OSGi Release 3
October 2005	OSGi Release 4
May 2007	OSGi Release 4.1
Sometime in 2009	OSGi Release 4.2

Table 1: OSGi framework history
Source: own table based on [60]

- Not enough tools for OSGi: There are so many tools available for OSGi. First remarkable milestone is Peter Kriens Bnd. It comes as a single JAR file. It is a tool for creating and diagnosing OSGi bundles. Using this tool one can view the manifest and contents of the bundles, create a bundle and can also verify available bundles manifest.
  Another remarkable tool for OSGi is SpringSources Bundlor. This tool is more or less equally powerful as Bnd tool but it overcome some of the limitations faced by Bnd tool.
  Apache Felix, one of the OSGi frameworks also has several sub-projects that provide OSGi tools. Most of them are runtime tools.

- OSGi is only for Eclipse users: At some extent it seems to be true that OSGi is for Eclipse users because many OSGi implementations are produced by Eclipse Foundation but it does not mean that it is dedicated for Eclipse only. Moreover, it is also a fact that Eclipse has provided the best IDE for OSGi till date.

2.4 Current OSGi framework implementations

Here are some of the current popular OSGi framework implementations, see Table 2.

Name	License
Apache Felix	Open source
Concierge OSGi	Open source
Equinox OSGi	Open source
Hitachi SuperJ	Commercial
Knopflerfish	Open source
ProSyst mBS	Commercial

Table 2: Current OSGi framework implementations
Source: own table based on [58]

- Apache Felix: Apache Felix [6] is a community, runs by Apache Software Foundation and it is continuously working on the implementation of OSGi R4 framework under Apache 2.0 licence. It is a cross platform implementation of OSGi framework release 4. The stable release is 4.0.3 which was released in July,2012. To run this we have to download the file `felix-framework-4.x.x.tar.gz` or `felix-framework-4.x.x.zip` compressed file which is around 750kb in size in compressed form.
  The Felix project is divided into sub-projects and each sub-project targets a fix OSGi framework implementation. Some of the sub-projects of Apache are: Config Admin [7], Dependency Manager [8], Event Admin [9] and File Install [10].

- Concierge OSGi: Concierge [24] is based on OSGi release 3 framework and is about 80kb in size. Due to its small size, it is most suitable for mobile and embedded devices because of the memory and processing capabilities in these devices are limited. When the resources are limited, Concierge is the best solution because it is highly optimized version of R3.
  Concierge uses OSGi service platform R3 implementation and currently running its 1.0.0 version as a stable release which was made available somewhere around April, 2009 [14]. It works under BSD Style Licence.

- Equinox OSGi: Equinox [16] is an Eclipse project which implements OSGi release 4 which is developed by Free and Open Source software community. It allows the developers to write their application and then allows them to convert that application in the pure bundle form using common services.
  The Equinox is running its 3.8.1 version as the stable one which was released in Sept, 2012 [17]. It was released under Eclipse Public Licence.

- Hitachi SuperJ: SuperJ Framework [31] is designed for home network and is totally based on the OSGi

R4 specification and provides simple remote-control service platform to remotely control, which includes start, stop, install and uninstall of Java applications.
Its domain includes: Home gateway, Car navigation/on-vehicle systems, Digital TV/Set top box, and Multifunction printer.
SuperJ Engine is optimized for M2M (machine to machine) devices based on the license of JavaME CDC from Oracle.

- Knopflerfish: Knopflerfish [22] is a non-profit organization maintained by Makewave, developing OSGi related materials. This project is based on OSGi R4 v4.2 framework specification.
  Currently it is running its beta release 4.0.0.beta-1 which is available since Apr,2013 [23] and has a stable version 3.5.0 released in Jan 2013.
  It has been released under BSD Style license.

- ProSyst mBS: ProSyst [28]is a German based software company which was from one of the first companies who joined OSGi Alliance and now completely working on OSGi technologies for developing, implementing and updating IoT/M2M applications.
  ProSyst set its focus on the development of OSGi related software and solutions such as Frameworks, Bundles, Remote Management Systems and OSGi tools for developers [27].

3 Bundles

The very first topic when starting with OSGi is Bundle, which is the basic unit of modularity. It contains JAR files and other resources that provide functionality to the end user.

3.1 What is a bundle

According to the OSGi specifications, bundle is a "unit of modularization" [42]. It contains Java classes and other resources which when combined together can provide functions to end user.
To be more precise, a bundle is a JAR file which contains resources, manifest file which describes the content of that JAR file and provide information about that bundle, it can also contains some additional information.
In short bundle= JAR + OSGi information [42]

3.2 Life cycle of a bundle

The figure, see figure 5 which shows the complete life-cycle of an OSGi bundle and different transitions between states.

- Installed: Once the installation is successfully completed the bundle reaches in `Installed state`.

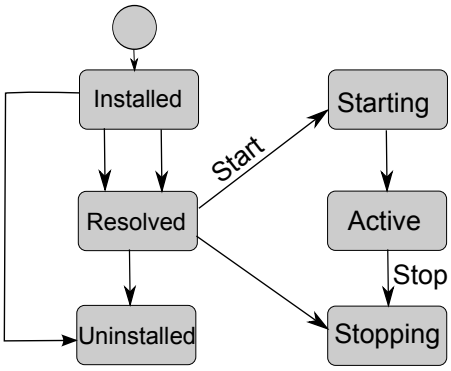


Figure 5: OSGi bundle life-cycle
Source: own figure based on [38]

- Resolved: A bundle can be in `Resolved` state only in two conditions, first, before the starting of the bundle and second, after stopping a bundle. This state means that all the dependencies which are required by the bundle are available.

- Starting: The bundle changes its state to `Starting` when the Activator of the bundle calls `start` method.

- Active: If the `start` method of the Activator of the bundle get executed successfully, the bundle jumps from `Starting` state to the `Active` state. More precisely, the bundle is running when it is in the `Active` state.

- Stopping: This state for a bundle is found whenever Activator of bundle calls the `stop` method.

- Uninstalled: When the bundle is removed from the system, this state is found. When a bundle is uninstalled from the system, it must be install again to access that again.

3.3 OSGi metadata

The OSGi metadata is represented in the JAR files by some header entries to tell the OSGi framework what the bundle provides or/and requires. There are around 20 manifest headers provided by framework. Here are some most commonly used and important headers:

- Export-package: In this section only those packages from the bundle which we want to be available to the outside world, are placed. It means those all are public and can be imported by other bundles. The packages which are not part of this are private by default and can not be accessed by the other bundles.
  `Export-Package: us.elron.bundles.ours.exported; version = "1.0.0"`

- Import-package: In this header section we specify the name of the packages that we are going to import. All those packages which are defined in this header will only be imported. If the imported package is not available, the bundle will fail to load.
  `Export-Package: us.elron.bundles.yours.exported; version = "1.3.0"`

- Bundle-SymbolicName: This is the only required header which is a unique identifier for the bundle to identify it among the pool of bundles.
  `Bundle-SymbolicName: us.elron.bundles.ours`

- Bundle-Name: This header defines the name of the bundle in more human friendly nature. It is a human readable name which can reflect more usefulness of that bundle, e.g., `Bundle-Name:Hello-World` conveys a message that this bundle deals with some `Hello World`.

- Bundle-Activator: The `Bundle-Activator` is an OSGi specific interface [55] which get called whenever a bundle is started or stopped. If `Bundle-Activator.Start` is executed successfully on a bundle then it can be guaranteed that `Bundle-Activator.Stop` will be executed automatically when that bundle is stopped. There are two methods associated with this interface. They are:
  `void start(BundleContext context)`
  Called when this bundle is started.
  `void stop(BundleContext context)`
  Called when this bundle is stopped.

- Bundle-Description: This is the description of the bundle. It may provide the information about the working of a particular bundle.
  `Bundle-Description: Very own bundle of ours`

- Bundle-Classpath: This header tells that from where to load the bundles. The bundles to be loaded may be in an uncompressed folder or it may be a JAR inside JAR. If it has to load bundles inside JAR files, it has to be extracted first into some temporary storage and it may take some additional time.

- Bundle-ManifestVersion; This header though purely optional but highly recommended to give. If this entry is not been give, it takes the default value 1. In OSGi R4, this has the value of 2, `Bundle-ManifestVersion:2`.

  Below is an example, taken from Spring 2.5.x core bundle manifest that uses some of the headers mentioned above, see Table 3.

Header Name	Value
Bundle-Name	spring-core
Bundle-SymbolicName	org.springframework.bundle.spring.core
Bundle-ManifestVersion	2
Export-Package	org.springframework.core.task;uses:=”org.springframework.core, org.springframework.util”;version=2.5.1 org.springframework.core.type;-uses:= org.springframework.core.annotation;version=2.5.1[...]
Import-Package	org.apache.commons.logging,- edu.emory.mathcs.backport.java.util.concurrent; - resolution:=optional[...]

Table 3: OSGi Metadata headers
Source: own table based on [42]

3.4 Package Versioning

When using OSGi it is always a good practice to explicitly mention the class with the proper version number which we are going to use, both when importing and exporting a package. It is because, in OSGi we can have multiple versions of same class. We can be achieve this through proper `versioning` . It is an attribute and can be added after each package declaration. The version format supported by OSGi is `<major>.<minor>.<micro>.<qualifier>` where major, minor and micro are numbers and qualifier is alphanumeric.
`<major>` indicates a significant update which doesn't guarantee any compatibility
`<minor>` indicates an update which preserves compatibility with older minor versions
`<micro>` represents an insignificant update from the user point of view which is perfectly compatible both forwards and backwards
`<qualifier>` is a user defined string it is not widely used and can provide an additional label to the version number, such as the build number or target platform without a standardized meaning

The default version (if the attribute is missing) is ”0.0.0”.

3.5 Working with OSGi Metadata using Bnd

Even after you have so much knowledge about bundles, tempering the bundle is not encouraged because any typos or extra space can make manifest useless. Manifest has its own format and no general text editor can understand that format and is not really able to give you the flexibility to play with these files. We have a very nice utility which can be used to create OSGi bundles and that tool is called Bnd [4] (stands for BuNDle tool), which is an open source utility created by Peter Kriens [3], OSGi technical officer. It is a very nice utility that ease the burden of programmer

by creating and diagnosing OSGi R4 bundles.
This `bnd.jar` file can be run by using command-line or it can be integrated in Eclipse as a plug-in.
Bnd can create jar from class files by adding necessary OSGi header information into them. It can also verify OSGi information from the existing jar files. These things make this utility quite simple and powerful.
The following line takes a vanilla.jar file as an input and after issuing a print command over this file with bnd.jar file, it displays the various general manifest information.
`-jar bnd.jar print c3p0-0.9.1.2.jar` [42]

The following line of code will create a new archive with same content as the original jar file but it will modify the `MANIFEST.MF` and save the achieve with a .jar extension.
`java -jar bnd.jar wrap c3p0-0.9.1.2.jar`

3.6 OSGi repository

A repository can be think as a central location which provides resources that can be used and in terms of OSGi, it is a central location or server where developers can put resources that can be downloaded and can be retrieved and installed on the system. The very good example is Linux [2] distribution which uses Advanced Packaging Tool (APT) [1] which helps in searching, installing or updating the software packages from repositories. At present we have so many existing repositories. These repositories have existing libraries wrapped inside the bundles which are ready to use by the programmer. Below are some existing repositories.

- OSGi Bundle Repository (OBR): OBR [26] provides a federated collection of bundles. It includes OSGi Alliance bundle repository. Some other companies which provide OBR repositories are Sonatype Nexus [30], Paremus [33] and Knopflerfish [21].

- Eclipse Orbit: The mission of Eclipse Orbit [25] project is to provide the libraries which are used in Eclipse project. It contains the old version of repositories from the different vendors and are easy to integrate by the developers.

- Apache Felix Commons: Apache Felix Commons [11] is a community dedicatedly working to create OSGi bundle version of different vendors open-source libraries, like Eclipse Orbit. The main aim of Felix is to avoid duplication in the libraries and sharing bundlise artefact.

- Apache OSGified projects: The Apache Felix project [13] made an effort for the Commons components to add the OSGi metadata into jar files so that it can be OSGi ready by adding OSGi entries in the jar's manifest. They made maven-bundle-plugin to ease this work.

4 OSGi and p2p network

Decentralization and p2p networks are connected to each other. In fact, decentralization is the back bone for the p2p systems. The study of these two key aspects is more important because our PG is also based on p2p systems. Let us first see what these topics mean.
Centralization in terms of computing refers when the most of the functions are carried out by some distant or remote centralized location. While on the other hand, decentralization means the resources (hardware and software) are allocated to some individual locations and these individual locations are responsible for all the functions to be processed. This approach has so many benefits over the normal centralized systems and provides better utilization of the resources. Peer-to-peer (P2P) [52] system is a distributed system in which a task is divided between individual machines or nodes which are called peers. These peers which may have different processing power and different resources available, take an active participation in the so formed network by making their resources available to other peers present in that network. These peers works as both, a producer and consumer of the resource. This approach provides us some very good advantages like, there is no need of administrator for these kind of network and network failure is decreased dramatically. These kind of systems are the basic building blocks of so many popular projects, e.g., BitTorrent.
OSGi framework works on the basic unit of modularity, called bundles. Under OSGi, the component updating is allowed for those packages which are available locally. In the other case if the required component is not available locally, OSGi fails to install or update it automatically. One solution to this problem is to use the OSGi Bundle Repository (OBR) but it lacks in the dynamic selection of the components. The other approach is to use an on-demand bundle selection and integration using peer-to-peer platform JXTA [51] and this platform is termed as decentralized dynamic code management (D^2CM). To use (D^2CM) we have to deal with the two technologies, namely, OSGi and JXTA.

4.1 JXTA

JXTA (Juxtapose) [56] is an open source peer-to-peer protocol specification begun by Sun Microsystems in 2001. Its been developed by Open Source Developers Community and is platform and programming language independent. This technology allows any connected device on the network to communicate with the other devices in p2p manner. The devices may be mobile phone or it may also be servers. It does not put any restriction on the type of the device or on the processing capabilities. Being programming language independent, this technology can be implemented using Java SE, C/C++ or by C#.

4.2 Decentralized dynamic code management platform

This section explains the D^{2}CM platform in detail. This section contains requirements, architecture and how we can integrate D^{2}CM with OSGi framework. For D^{2}CM technology to work, we use JXTA [61] technology with OSGi.

- Architecture: D^{2}CM platform architecture is shown in figure 6. It has three main components JXTA services, Bundle definitions and Platform services.

  JXTA Services [34]: It contains two main services.

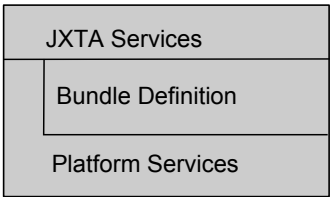


Figure 6: D^{2}CM architecture
Source: own figure based on [51]

They are code sharing service and code discovery. Code sharing services are responsible for loading and sharing for a particular resource while code discovery services are responsible for discovering and publishing resources.

Bundle definitions: According to OSGi specification, bundles contain some additional header information to work under OSGi framework. To make a bundle work under D^{2}CM platform, some more additional details are added to the manifest to make it dynamic. Thus it includes (`interface descriptive manifest` for IDA[4] and (`code descriptive manifest`) for CDA[5].

Platform services: The services are divided into three parts, see figure 7. The loading services are responsible for automatic code selection and loading while repository services unload the services for the local bundles and these bundles are automatically resolved by resolver services.

- Requirement: The functionality of the component should be defined clearly. One can use any platform and programming language independent notation, see figure 8. This figure shows the description of a bundle. It contains many attributes which describes different aspects of the functionality of the bundle, e.g., `functional-properties` may tell that what are the actual functions that bundle has to contain or to provide.

[4]Interface Description Advertisement-used to announce the existence of an interface

[5]Code Description Advertisement-used to describe implementation of an interface

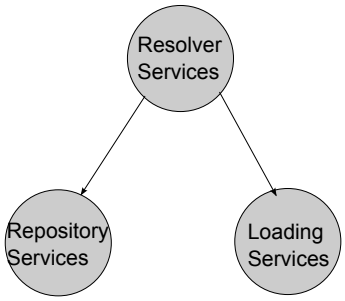


Figure 7: D^{2}CM OSGi services
Source: own figure based on [51]

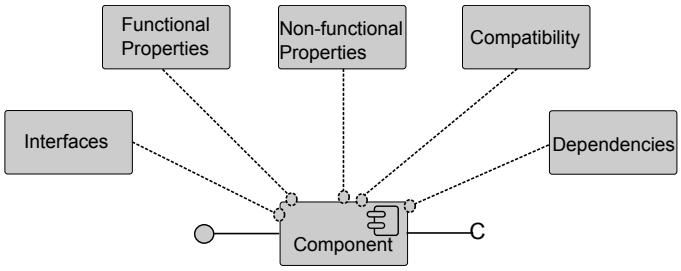


Figure 8: Description of OSGi bundles
Source: own figure based on [51]

For dynamic bundle loading, this platform has to provide so much other additional functionalities. The figure 9 shows those additional functions that this platform has to provide. This includes discovery, selection, transfer and dependencies resolving of the bundles.

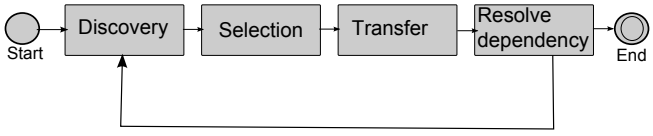


Figure 9: OSGi bundle discovery and loading
Source: own figure based on [51]

- Integrating with OSGi framework: We know that Equinox and Apache Felix are some popular OSGi framework implementations. Further, they provide a console for managing OSGi bundles. This D^{2}CM platform is also integrated with these consoles. The command `dcm_resolve <bundle_id>`, where `bundle-id` is the id of the bundle to be resolved, is used for automatically resolving an installed bundle. Internally the resolver service of the framework

runs and provide list of bundles for the dependency. Moreover, while installing we can enter the command `dcm_install <url>` for automatically resolving the dependency.

4.3 Decentralized service discovery

Service Oriented Computing (SOC) is backbone for distributed applications. To gain the full advantage from SOC, it must provide reliable, secure and scalable service discovery mechanism [40]. Traditional service discovery mechanism which uses centralized web server approach have been seen to suffer from performance and they are more prone to the failures, as the number of users grow, the demand also keep on rising. Hence centralized web servers are not the perfect choice for a large scalable service network. As it is clear that SOC is based on distributed environment, this approach can be used in the conditions effectively where centralized approach fails.
This section explains Chord4S, a Chord-based decentralized service discovery approach that supports service description and discovery in p2p manner [40].

4.3.1 Service discovery

- Centralized service discovery: Centralized client/server model has been used since the foundation of SOC for service discovery in conjunction with UDDI (Universal Description, Discovery and Integration) [37]. In the current age also, UDDI is been used by so many software vendors in their software to provide efficient, reliable and scalable service discovery mechanism. But more or less, every centralized approach unfortunately cannot provide these all things together.

- Decentralized service discovery: Decentralized service discovery approach provides a better solution to the problems suffered by centralized service discovery mechanism. There has been so many efforts putted on the field to make p2p as a base of service discovery. For example, Web Service Dynamic Discovery (WS-Discovery) [35] is a multicast discovery protocol in which to locate a service, a request is send to the multicast group.

4.3.2 Service description

The service description of Chord4S is mainly divided into three parts. They are service identifier, QoS specification, and syntax specification.

- Service identifier: Service identifier under Chord4S is divided into two main parts, function bits and provider bits. The main function of the function bits are to provide the service and the provider specific information is provided by the provider bits. The service identifier is layered and thus it makes it hierarchical in nature. For example a five layer service description may contain four layers of functional bits and one layer for the provider bits. In certain cases, four layer of service description is also acceptable but in that case the fourth layer for the function bits will be replaced by the zeros.

- QoS specification: QoS specification are the most crucial aspects in the service discovery because of the fact that QoS requirement for each and every consumer differs from one another. So it has to return only those services that meets the requirement of that particular consumer. Under Chord4S, the consumer has to first check the service identifier, if it matches the functional requirements which that consumer is interested in, he can further go and check the quality specification which are attached. Chord4S supports three kinds of QoS attributes, Numeric QoS Attribute, Boolean QoS Attribute and Enumerated Type.

- Syntax specification: Syntax specification, as the name implies deals with the syntax of the service in which a particular consumer is interested. It mainly checks the input and output parameters of the service. The input and output parameters must match while providing and using a service from the pool of available services. These all tasks are the responsibility of the syntax specification defined under Chord4S.

4.3.3 Service publication in Chord and Chord4S

Service publication is an important part in service discovery in decentralized systems. This section explains how the Chord4S service publication method is different from the traditional Chord service publication.

- Service publication in Chord: Chord is a distributed lookup protocol [53] that solves the problem of service discovery in a pure p2p network. This protocol is capable enough to find a particular node which has the specific data item a consumer is looking for. When a key is provided to Chord and this protocol discover that node which matches the given key. It uses DHT (Distributed Hash Table) to maintain a key/data item pair. Chord is so flexible and dynamic in nature and it can adapt itself according to the change in the number of nodes in the network, as it is evident that in p2p system, any node can go offline or new nodes can join anytime.

- Service publication in Chord4S: Chord4S, though is implemented on Chord but it overcome the problems faced by traditional approaches, namely, data replication and redundancy. Chord4S introduces a new

concept of functionally equivalent node. It means, it identifies the nodes which are functionally equivalent to the node which has the data which is required by the consumer. Once the node that has the particular data, is down due to any reason, the consumer can contact the functionally equivalent node for that particular data/service. This technique avoids the data replication. To avoid redundancy, some further changes has to introduce into design specification.

4.3.4 Service query types in Chord4S

Chord4S is a protocol to provide service discovery. To provide service discovery it has to query the network and the query must route from different nodes. This section presents query routing in Chord4S.

- Query types: Two types of query is supported by Chord4S. They are, Service Specific Queries and Queries with Wildcard(s).
  Service specific queries are the simple one and very specific. In these types of queries, the consumer has to tell all the specific details and put these details into the layers. These information then transferred as a query. These all layers get hashed and the resulting bits forms target specific identifier.
  Queries with wildcard(s) is the complex type of queries and are fired when the consumer is looking for a bunch of services. It may be the case that the consumer is looking for all related things. It has the same concept in terms of working as service specific queries. It also store its details onto the layers, but all those layers which have wildcard(s), their corresponding hash is replaced by zero.

- Query forwarding: Traditional Chord does not support getting the list of multiple matched service. Whenever a service is matched in Chord, the query stops to that node and does not propagate further, hence it limits itself in producing the better service results. In Chord4S this limitation has been overcome. It means even after the first service match, the query fired under Chord4S does not stops and it continues to propagate inside the circle until and unless it finds the sufficient number of matched nodes.
  Forwarding service specific queries: This type of query contains a counter and a target service identifier, whereas target service identifier contains function bits and provider bits but in this case provider bits are all zeros. This target service identifier is matcher with each service identifier. To detect a match, we perform bitwise AND of these two. If the resultant bit pattern is exactly same as the target service identifier, we call it a match. After this, the same query is forwarded for finding more providers of the same service.
  Forwarding queries with wildcard(s): This type of query also works exactly similar to that of forwarding service specific queries except a small difference. In forwarding queries with wildcard(s), we consider only those bits for matching which provide explicit service information.

4.3.5 Comparison between Chord and Chord4S

This section compare Chord with Chord4s. This section deals with the experimental results produced on the Chord simulator for evaluating the performance of Chord4S. Two important aspects have been analysed in this section, which are data availability and routing performance. Please note that for this section all images are taken from [40] as it is.

- Data availability: To measure this, the performance has been recorded for the percentage of fail queries to the failed node. The nodes are chosen to be volatile and using these kind of nodes the percentage of fail queries have been figured out.

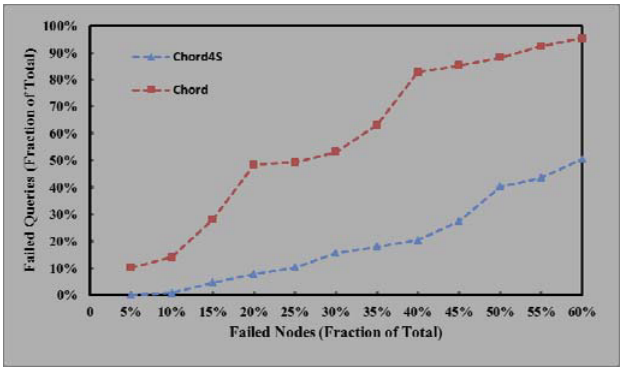


Figure 10: Chord4S data availability for 2^7 nodes
Source: taken from [40]

See figure 10 which shows this set of experiment, organized for a network of 2^7 nodes.

See figure 11 which shows this set of experiment organized, for a network of 2^{11} nodes.

See figure 12 which shows this set of experiment, organized for a network of 2^{15} nodes.

- Routing performance: This performance is measured on the basis of the number of hopes taken for that query to service the demand. See figure 13 to see the routing performance based to services specific queries for 2^7, 2^{11} and 2^{15} nodes.
  See figure 13 to see the routing performance based to queries with wildcard(s) for 2^7, 2^{11} and 2^{15} nodes.

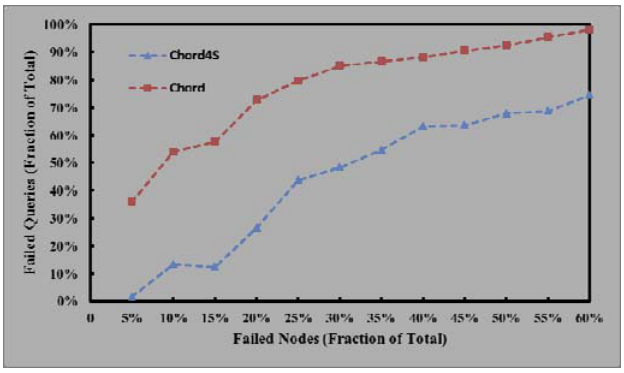


Figure 11: Chord4S data availability for 2^{11} nodes
Source: taken from [40]

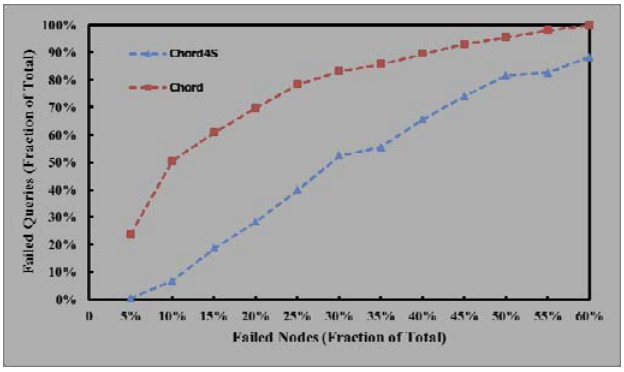


Figure 12: Chord4S data availability for 2^{15} nodes
Source: taken from [40]

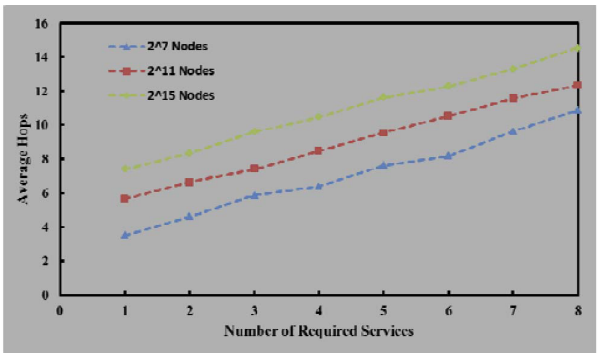


Figure 13: Routing performance for service specific queries
Source: taken from [40]

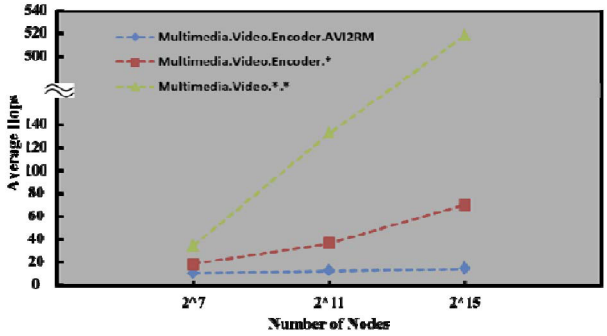


Figure 14: Routing performance for queries with wildcard(s)
Source: taken from [40]

4.4 A survey on resource discovery mechanism

A very important aspect in today's growing network size, is the discovery mechanism for service and resource. The service and resource discovery is becoming a very challenging field because of the addition of mobile and wireless networks [46]. Though it is a very vast field and it has so many technologies available these days, this field is continuously increasing in the protocols and frameworks for providing better solutions for discovery. In this section we will study resource and service discovery in general but due to the complexity of the topic and research made in this field, we will limit our focus to the resource discovery in decentralized and p2p systems only.

4.4.1 Enabling technique in service discovery (SD)

In this section, the main aim is to provide sufficient knowledge about the different techniques available for packet forwarding and information sending.

- Packet propagation: A cleverly selected packet propagation technique not only provide better service discovery but also minimise the time taken for that. These following four techniques are most widely used for packet propagation. They are:
  Unicast: The simple technique that involves only one sender and one receiver.
  Broadcast: In this protocol there is one sender and many receivers.
  Multicast: In this protocol a sender sends packet to group of nodes with single send operation.
  Anycast: It is as similar to multicast but it differs slightly, as the packet is delivered to the node that best describes the given parameters.

- Relative Vs proactive announcement (Push Vs Pull): The user demands service and can get the service in two ways, either user searches that service explicitly or it can come to know about this via some advertisement. If the user searches the service and then discovers it, it is called reactive approach, which is the underlying principle for Gnutella2 and eDonkey. On the other hand if the user came to know about a service via advertisement, the so called approach is called proactive, e.g., Bonjour is based on proactive approach.

- Other ways for information delivery: Some additional methods for propagation is discussed below.
  Caching: Caching is a very popular method which can be used by the intermediate nodes to store the result of the query and hence it can reduce the traffic considerably. But the care must be taken with this approach because it may be possible that the cache is containing some outdated values.
  Regular updates: To keep the accurate information into the cache, regular updates are employed. It is true that over a period of time, the cache may contain some outdated value and hence it needs to get updated after some predefined interval. But again some measure should be taken care of in regular updates so that it should not impose the overhead on the network in this operation.
  Tracing: When in a network if some result is found on some node, tracing message is generated. This message takes the same path back to the source and it provides the information about the nodes which are visited but this method does not provide any kind of guarantee that if the message will reach to the node it has been dispatched for.
  Piggybacking: Whenever an additional message is imposed on the outgoing packet, piggybacking arises. Piggybacking does not increase the traffic because it has been imposed on the outgoing packet, however, due to imposing some extra details, the outgoing packet might increase in size. Practically we are not using the channel twice but we are using channel once but a slightly increased size package. It suffers with the problem that it might be possible that the message will not reach to the desired node.

- Dynamic query termination methods: In the resource discovery, one crucial thing is that the search query should not die before the minimum number of hopes to ensure better service discovery. The another key aspect is the radius that a query has to cover for search. We have three methods for ensuring that query can travel a certain distance. They are:
  Iterative deepening: Combined with flooding based systems, with each iteration it covers larger network map. The query dies either when the searched item is found or the query timeout expires.
  Checking: In this method the neighbour nodes are chosen in random and the query is forwarded to these randomly selected nodes only. The query continues to go forward in this manner until its TTL hop-count expires.
  Chasing wave: This technique is based on chasing game [45] but in this case the packets are chased. We have to stop the replicas by chasing them on the network whenever the query is answered from anywhere.

4.4.2 DHT based system in resource discovery

Distributed Hash Table is a technique which allows the system to find for a particular item when provided the key of that particular item as an input. This technique provides decentralized and fully scalable systems but on the other hands they suffer with the drawback that they cannot support much complex queries. For this to operate, the queries must be simple. In this section we will enlighten some of the decentralized structured architectures based on DHT. They are:

- Ring topology (Chord): The main idea in this topology is to forward package in one direction only, just like a circle/ring. The packets are forwarded in the ring until they reach to the destination. Because it works in a ring manner, only require thing that one node should know is its successor.

- Tree topology (Plaxton [48], Tapestry [62]): In this topology the key value is compared to the id of the node, if more than one bit forms a match (prefix only), then the packet is forwarded to that particular node.

- Multi-dimensional cartesian space topology (CAN): In Content Addressable Network (CAN) [49] the nodes are divided into logical area. This approach is a greedy approach and the packet will be transferred to the node which seems to be more closer to the destination node. For this topology to work, each node must know the coordinates of its neighbour and their IP addresses.

- Kademlia/XOR topology: Kademlia [44] is XOR based decentralized p2p system. A total of 160 bits are used to node address and key. This information is stored to all the nodes who appears to be close after calculating the XOR metric (Distance= key XOR NodeId) and the same calculation is carried out while looking for a key value in the network.

- Viceroy/Butterfly [43] topology: This topology creates a logarithmic diameter for creating a connection graph. This topology is exactly similar to that of Chord with a slight difference, the nodes in Viceroy maintain both the successor and predecessor in the ring, hence saves considerable amount of time.

- Pastry/Hybrid topology: This is pure hybrid topology. The search made in Pastry [50] can take one of the two approaches, namely, Ring or Tree. For this to implement, Pastry maintains three tables. One table is for routing process, second is leaf table, which is used for ring routing and the third is table to maintain its neighbours. A good news about Pastry is that it assures that it will not fail until half of its leaves are fail simultaneously.

4.4.3 Resource discovery in p2p

This section gives an overview of some popular p2p systems.

- Napster [5]: The main aim of this p2p system was to share MP3 files. It was based on simple client/server architecture and responsibility of the server included to have a list of resources its client has. The basic mechanism of Napster was that the user fires a query for the desired file to the server, server searches that item in the list that it maintains, meanwhile the user waits. After sometime the server replies back to the user and points the node from where the user can get the desired file. Due to legal issues, unfortunately Napster was forced to shut down.

- Gnutella [18] and Gnutella2 [19]: Next in the line of p2p systems after Napster was Gnutella [36] which was decentralized and based on flooding for file search. Though it suffered with so many drawbacks but it was the base for the second version of it, namely Gnutella2. In this the nodes were named as leaves and hubs. When the user searches for some file, this search is sent to the hub. In the case if the file is found on this node, it is downloaded directly but in case of a miss the query forwarded to the next hub and so on.

- KaZaA and FastTrack: KaZaA [20] is a file client which connects to the FastTrack [15] p2p file sharing network. It employs two level architecture, on level 0 we have normal nodes and on level 1 we have super-peers. The super-peers maintains the list of the resources that the normal nodes have. Whenever a node fires a search query for the particular file, it is made to the super-peers only. The problem with this system was that users encountered too much fake files.

- eDonkey [54]: This system is based on eDonkey2000 protocol. In this we have a client/server architecture and the server maintains the list of file that its client has. Further, the server contains no files itself but it contains only the list of resources provided by its clients. For this system to works, initially server sends its users a list of other servers, before which the file search is not possible.

- Freenet [32]: Freenet is a perfect example of unstructured p2p system. Its working is also quite simple. Whenever a file is added in the system, the system assigned a globally unique identifier (GUID) which uniquely defines this file in the network. The content hash is calculated to generate content-hash which are used to generate GUID for this file. Whenever the search is fired by any user, it get forwarded by the intermediate nodes to the node which it thinks are more closer to the node that may contain this file. Whenever a file is found its replica is made in the network path, hence provides a better search for the most popular files. In other words, the most popular files are likely to be found very easy.

- BitTorrent [12]: BitTorrent is undoubtedly is the most popular and fast third generation p2p system. It hosts the list of files at a central location with the extension of .torrent which the user has to download to get that required file. The user searches for the particular file on the central database, the server gets the list of all files matching search criteria. The user download that torrent file and after adding this file the actual download of the file starts. BitTorrent supports download from the multiple source at the same time. The file is split up into smaller pieces and those pieces are shared by all the users who are interested in that particular file. Unfortunately, it also suffers from some legal issues.

4.4.4 Some service discovery frameworks

This section gives the knowledge of some popular service discovery frameworks.

- SLP: Service Location Protocol [29] was designed to provide service discovery for the wide variety of networks. It works in two modes, first one is a centralized mode in which we have a discovery agent and all the service requests and service replies flows from the discovery agent. In the second mode, there is no discovery agent but it uses multicast to find a service and gets a unicast reply.

- Jini: Jini framework is implemented in Java so it is platform independent, as long as JVM can be installed, this framework is going to work. It is very flexible in nature too. Since Jini framework is based on Java, all the attributes and services are represented by classes and objects only. The main part in Jini [41] framework is `lookup service` and it is same as the discovery agent in SLP.

- Service discovery in bluetooth: Service discovery in bluetooth devices is a perfect example of resource constraint environment where devices that has to communicate must be within a short range. When a device wants to send some file it performs service discovery and sends a signal to all nodes in its range. The node which wish to communicate, send their details to the device.

- JXTA: JXTA (Juxtapose) [56] is an open source peer-to-peer protocol specification begun by Sun Microsystems in 2001. This technology allows any connected

device on the network to communicate with the other devices in p2p manner. The devices may be mobile phone or it may also be servers. It does not put any restriction on the type of the device or on the processing capabilities.

4.4.5 Some challenges in SD

Service discovery is not an easy task to achieve because each type of environment suffers with some problem. This section explains some of the challenges faced in SD.

- Importance of network topology in SD: The particular topology employed for the SD plays a vital role in terms of the performance. For example, comparing the Chord topology which works on the ring topology with the Viceroy topology which is an extension of the Chord with the difference that in this the packets can move to either left or right in a ring. So in certain cases, the performance of the Viceroy topology is much better than that of Chord.

- Overlay topology construction & topology mismatch: The overlay network never maps with that of the physical topology so it can produce some mismatch and performance degradation. The search and route mechanism of DHT based system is far away in the working with that of physical topologies.

- Bootstrapping: Bootstrapping happens whenever a new device is connected to a network. For this node to be a part of service discovery network, must announce its presence in the network. This may be by simple flooding in case of small networks or it may be use a central registry server. When a node is registered into a network, it stored the addresses of other nodes by searching the network. When after sometime this node leaves the network and rejoins after sometime, it still has the old addresses and again acquires the same address one more time. It increases redundancy in the overlay and costs precious traffic in the network.

- Support for more sophisticated queries: The queries that are supported in SD mechanism are simple one which are based on keywords. It has to extend the capabilities to support more dynamic and complex queries to include range and relational queries.

5 Conclusion and future work

In this paper we have seen that under OSGi framework, JAR files can be used beyond its capabilities and this framework can be used for creating bundles and repositories. We have seen that using OSGi is so much beneficial and economical and only few additional entries are required to be added to the JAR files to use it under this framework. The bundles are the unit of modularization under OSGi framework and it contains JAR file with some OSGi metadata in which only `Bundle-SymolicName` is the only required field to provide. Using this framework the problem of JAR-hell can also be solved. These all things makes OSGi a very important part to include in our PG.

The only problem with the OSGi is it fails in loading the bundles which are not available locally and to overcome this problem a new technology called D^2CM (Decentralized Dynamic Code Management) is used which combined OSGi and JXTA (a protocol which allows the devices in a network to communicate in p2p manner). We studied the architecture of JXTA and have seen that to make good use of D^2CM (Decentralized Dynamic Code Management) technology it must be integrated it with the inbuilt console provided by Equinox and Apache Felix.

As the current PG is based on p2p systems and a very important part in p2p system is the service discovery. We studied two decentralized service discovery approach which are Chord and Chord4S. Chord is capable enough of finding any node when given a key is provided. It is a DHT based system. Another approach which is also based on Chord is Chord4S, which introduces the concept of functionally equivalent node hence it is a perfect approach that can be used in our current approach as this approach can reduce data replication in the network which can be a very challenging task in p2p system. On the other hand, Chord4S supports in getting the list of multiple matched services while Chord fails in this case which also makes Chord4S a better approach to be used in PG. We also have been seen some experimental results that clearly supports that Chord4S is a better approach that Chord in terms of data availability and routing performance, see section 4.2.5.

We have also seen some very popular p2p systems like Napster and BitTorrent and how they work, what are the underlying technologies for the file transfer. To integrate file transfer in our PG, we can use Freenet so that the most popular file is more likely to be found but it increases the overhead of replication in the network. The technology of BitTorrent can also be a good approach where we can make a special kind of file which contains all the information but to host these files we must find some kind of server or must figure out some other way so that user can search for a particular file. It may be possible to choose some nodes randomly as the server but nodes can leave anytime from the network which is again a big problem.

References

[1] Apt-Debian Wiki. `http://wiki.debian.org/Apt`, 2003. [Online; accessed 12-Jun-2013].

[2] Linux.org—Resouces for Linux—How to Linux Guide—What is Linux. `http://www.linux.org`, 2003. [Online; accessed 12-Jun-2013].

[3] aQute—Bnd. http://www.aqute.biz/Bnd/Bnd, 2006. [Online; accessed 22-May-2013].

[4] BndTools. http://bndtools.org/, 2011. [Online; accessed 22-May-2013].

[5] Napster. http://www.napster.de/start, 2011. [Online; accessed 22-May-2013].

[6] Apache Felix. http://felix.apache.org/, 2013. [Online; accessed 22-April-2013].

[7] Apache Felix-Apache Felix Config Admin. http://felix.apache.org/documentation/subprojects/apache-felix-config-admin.html, 2013. [Online; accessed 22-April-2013].

[8] Apache Felix-Apache Felix Dependency Manager. http://felix.apache.org/documentation/subprojects/apache-felix-dependency-manager.html, 2013. [Online; accessed 22-April-2013].

[9] Apache Felix-Apache Felix Event Admin. http://felix.apache.org/documentation/subprojects/apache-felix-event-admin.html, 2013. [Online; accessed 22-April-2013].

[10] Apache Felix-Apache Felix File Install. http://felix.apache.org/documentation/subprojects/apache-felix-file-install.html, 2013. [Online; accessed 22-April-2013].

[11] Apache Felix Commons. http://felix.apache.org/site/apache-felix-commons.html, 2013. [Online; accessed 22-May-2013].

[12] BitTorrent p2p System. http://bittorrent.com/index.html, 2013. [Online; accessed 28-May-2013].

[13] CommonsOSGi-Commons Wiki. http://wiki.apache.org/commons/CommonsOsgi, 2013. [Online; accessed 22-May-2013].

[14] Concierge OSGi-Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Concierge_OSGi, 2013. [Online; accessed 22-May-2013].

[15] Content of FastTrack Protocol. http://cvs.berlios.de/cgi-bin/viewcvs.cgi/gift-fasttrack/giFT-FastTrack/PROTOCOL?view=markup&content-type=text%2Fvnd.viewcvs-markup&revision=HEAD, 2013. [Online; accessed 22-May-2013].

[16] Equinox. http://www.eclipse.org/equinox/, 2013. [Online; accessed 22-May-2013].

[17] Equinox(OSGi)-Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Equinox_OSGi, 2013. [Online; accessed 22-May-2013].

[18] Gnutella-Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Gnutella, 2013. [Online; accessed 22-May-2013].

[19] Gnutella2-Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Gnutella2, 2013. [Online; accessed 22-May-2013].

[20] Kazaa-Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Kazaa, 2013. [Online; accessed 22-May-2013].

[21] Knopflerfish Bundle Repository 3.5.0. http://www.knopflerfish.org/repo/, 2013. [Online; accessed 22-May-2013].

[22] Knopflerfish OSGi-Open soruce OSGi service platform. http://www.knopflerfish.org/index.html, 2013. [Online; accessed 22-May-2013].

[23] Knopflerfish-Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Knopflerfish, 2013. [Online; accessed 22-May-2013].

[24] Maven-Concierge OSGi, An optimized OSGi R3 implementation for mobile and embedded systems. http://concierge.sourceforge.net/, 2013. [Online; accessed 22-April-2013].

[25] Orbit. http://www.eclipse.org/orbit/, 2013. [Online; accessed 22-May-2013].

[26] OSGi Alliance— Repository/OSGi Bundle Repository (OBR). http://www.osgi.org/Repository/HomePage, 2013. [Online; accessed 22-May-2013].

[27] Prosyst-Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/ProSyst, 2013. [Online; accessed 22-May-2013].

[28] Prosyst-Your partner for Open Standard Based Middleware, OSGi. http://www.prosyst.com/startseite/, 2013. [Online; accessed 22-May-2013].

[29] Service Location Protocol-Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Service_Location_Protocol, 2013. [Online; accessed 28-May-2013].

[30] Sonatype Nexus. http://www.sonatype.com/nexus/features, 2013. [Online; accessed 22-May-2013].

[31] SuperJ Engine— Hitachi Solutions Ltd. http://www.hitachi-solutions.com/superj/sp/sje/, 2013. [Online; accessed 22-May-2013].

[32] The Freenet Project. https://freenetproject.org/, 2013. [Online; accessed 22-May-2013].

[33] The Private Cloud runtime for composite applications - Paremus Service Fabric. http://www.paremus.com/, 2013. [Online; accessed 22-May-2013].

[34] J. Arnedo-Moreno and M. Domingo-Prieto. An anonymity layer for jxta services. In *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, pages 102–107, 2011.

[35] J. Beatty, G. Kakivaya, D. Kemp, T. Kuehnel, B. Lovering, B. Roe, C. S. John, J. Schlimmer, G. Simonnet, D. Walter, J. Weast, Y. Yarmosh, and P. Yendluri. Web services dynamic discovery (ws-discovery). *http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf*, 2005.

[36] J. E. Berkes. Decentralized Peer-to-Peer Network Architecture:Gnutella and Freenet. http://www.berkes.ca/archive/berkes_gnutella_freenet.pdf, 2003. [Online; accessed 22-May-2013].

[37] L. Clement, A. Hately, C. von Riegen, and T. Rogers. UDDI Version 3.0.2. http://www.uddi.org/pubs/uddi_v3.htm, 2004. [Online; accessed 22-May-2013].

[38] Elron. OSGi: An Introduction. http://www.javacodegeeks.com/2012/01/osgi-introduction.html/, 2011. [Online; accessed 28-April-2013].

[39] R. Hall and H. Cervantes. An osgi implementation and experience report. In *Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE*, pages 394–399, 2004.

[40] Q. He, J. Yan, Y. Yang, R. Kowalczyk, and H. Jin. A decentralized service discovery approach on peer-to-peer networks. *Services Computing, IEEE Transactions on*, 6(1):64–75, 2013.

[41] M. Kelaskar, V. Matossian, P. Mehra, D. Paul, and M. Parashar. A study of discovery mechanisms for peer-to-peer applications. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '02, pages 444–, Washington, DC, USA, 2002. IEEE Computer Society.

[42] C. Leau. Creating OSGi Bundles. `http://blog.springsource.com/2008/02/18/creating-osgi-bundles//`, 2008. [Online; accessed 27-April-2013].

[43] D. Malkhi, M. Naor, and D. Ratajzcak. Viceroy: Scalable emulation of butterfly networks for distributed hash tables, 2003.

[44] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *Proceedings of the first international workshop on Peer-to-Peer systems*, IPTPS02, page 263, 2002.

[45] E. Meshkova, J. Riihijrvi, and P. Mhnen. Evaluation of dynamic query abolishment methods in heterogeneous networks.

[46] E. Meshkova, J. Riihijrvi, M. Petrova, and P. Mhnen. A survey in resource discovery mechanism. 2008.

[47] P. Paulin. Why is OSGi important? `http://java.dzone.com/articles/why-osgi-important`, 2009. [Online; accessed 22-May-2013].

[48] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures*, SPAA '97, pages 311–320, New York, NY, USA, 1997. ACM.

[49] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, pages 161–172, New York, NY, USA, 2001. ACM.

[50] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, Middleware '01, pages 329–350, London, UK, UK, 2001. Springer-Verlag.

[51] H. Schmidt, J. H. Yip, F. J. Hauck, and R. Kapitza. Decentralised dynamic code management for osgi. 2008.

[52] R. Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pages 101–102, 2001.

[53] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, California, August 2001.

[54] D. Stutzbach, R. Rejaie, and S. Sen. Characterizing unstructured overlay topologies in modern p2p file-sharing systems. *Networking, IEEE/ACM Transactions on*, 16(2):267–280, 2008.

[55] Unknown. Interface BundleActivator, OSGi Service Platform Core Specification Release 4 Version 4.3. `http://www.osgi.org/javadoc/r4v43/core/org/osgi/framework/BundleActivator.html//`. [Online; accessed 14-May-2013].

[56] Unknown. JXTA. `http://en.wikipedia.org/wiki/JXTA//`. [Online; accessed 21-May-2013].

[57] Unknown. Java Classloader-Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Java_Classloader`, 2011. [Online; accessed 22-May-2013].

[58] Unknown. OSGi. `http://www.en.wikipedia.org/wiki/Osgi/`, 2013. [Online; accessed 22-April-2013].

[59] Unknown. The OSGi Architecture. `http://www.osgi.org-Technology/WhatIsOSGi/`, 2013. [Online; accessed 27-April-2013].

[60] C. Walls. A Dozen OSGi Myths and Misconceptions. `http://www.jroller.com/habuma/entry/a_dozen_osgi_myths_and`, 2009. [Online; accessed 27-April-2013].

[61] H. Wang and Z. Deng. Research on jxta architecture based on mobile agent. In *Genetic and Evolutionary Computing, 2009. WGEC '09. 3rd International Conference on*, pages 758–761, 2009.

[62] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz. Tapestry: a resilient global-scale overlay for service deployment. *Selected Areas in Communications, IEEE Journal on*, 22(1):41–53, 2004.

Conceptualization of Online Social Links in Peer-to-Peer Overlays

Prince Singh
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email:princes@mail.upb.de

Abstract

This paper is focused on the prerequisites for development of an structured Peer-to-Peer Overlay based on Key Based Routing API of Online Social Networks. For creation of such a content based distribution system study has been done on the functionality of key features and the principles of Online Social Networks. A intense light has been thrown on Location Based Services or geographical positioning of the nodes in Peer-to-Peer Overlay System.

1 Introduction

In the modern era of globalization Peer-to-Peer Online social networks has emerged as a boon in the field of data sharing with reliability in the networks. Online social networks have bounded the world into one small screen.These online social networks are commonly used among the friends rather than the unknown world and the usage of such network is also not complex.

This structured Peer to Peer Overlay system has been found as a core for the foundation of large scale distributed system.*Structured overlays conform to a specific graph structure that allows them to locate objects by exchanging O(lg N) messages where N is the number of nodes in the overlay* [2]. describes that, these overlays are very beneficial for construction of distributed hash tables, scalable group multicast and locations of centralized objects.Services provided by the structured overlays are vast ,rapid, reliable, measurable and supports distributed data applications.Different types of API are supported by structured overlays which gives challenge to the designers for better understanding of the complicated parts of the overlay.For better functioning of the structured overlays applications they must be placed into the perfect system which in turn gives maximum functionality.Applications in the system are completely isolated and can not provide solutions for new requirements in various protocols.The difference's in the structure designs make's *comparative evaluation* of various other protocol structures hard.

2 Background

In this part, a brief description of common functionality among the structured or centralized overlays system protocols is provided. The *node* is predicted as an platform for the overlay participation. Various nodes can be represented by a common IP address in the overlay system as graphed in Figure 1.The identifier zone or space provides unique *nodeIDs* to every single node which is participating in the structured overlay system.Objects belonging to a particular application and sharing the same identification region are provided with unique identifications named as *keys*.Circular identification space is used by Tapestry, Chord and Pastry of n-bit integers modulo 2n (n = 160 for Chord a nd Tapestry, n = 128 for Pastry) as described by[1].128-bit nodeIDs are used by CAN which represents it in the identifier zone.

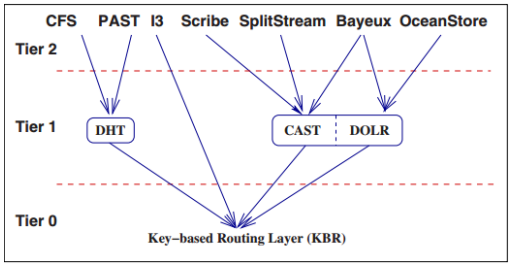


Figure 1. Basic abstractions and APIs, including Tier 1 interfaces : Distributed hash tables (DHT), decentralized object location and routing (DOLR) and group anycast and multicast (CAST) [2]

The working principle of overlays is based on Key's root or unique live node.All the keys are dynamically linked to the overlay system by key root.Routing tables are maintained for passing of the messages safely between the nodes.These routing table consist of node Id's and IP addresses to which nodes are linked in the overlay system.Messages are delivered to the nodes from the overlay links which are nearer to the key in identifica-

Table 1. Tier 1 Interfaces

DHT	DOLR	CAST
put (key, data)	publish (objectId)	join(groupId)
remove (key)	unpublish (objectId)	leave(groupId)
value = get (key)	sendToObj (msg, objectId, [n])	multicast (msg, groupId) anycast (msg, groupId)

tion zone.All the functions are defined by the system and keys are linked to the corresponding nodes.Keys are connected clockwise to those nodes whose node ID is nearest to the key in Chord.Pastry, is similar to the Chord except that no clockwise connections are done to nodes.It uses the simple method i.e to connect the keys to the live node whose node ID is nearest to the key.Tapestry follows longest prefix matching technique to connect the keys to the live nodes and node ID value for every digit is taken and which cannot be similar exactly.CAN,has a different method of working and follows the strategy of partitioning.Here every node divides the identifier space surroundings to its nodeID's.Furthermore, all the keys are linked nodes in the partitioned space that holds keys.

3 Prerequisites for Overlay Network

The foundation of the current system is laid on the basic structured overlays which also supports higher level abstractions.According to [1] examples for such systems are Distributed Hash Tables (DHT), Decentralized Object Location and Routing (DOLR), and group anycast/multicast (CAST). table 3 clears the concept of relations among these abstractions layers.*Teir 0* is the only layer in which all the Key Based Routing is supported by all the systems.*Tier 1* at this level which is above *Tier 0* and at this layer only few current systems provides high level abstractions.Usage of these abstraction is one or more in most of the cases by *Tier 2 services* or applications.In some cases Key Based Routing is directly achieved example i3.

The key functions of the abstraction layers such as Tier 1 ,Tier 0 and Tier 2 are defined in the table above.The working principle of the DHT is similar to traditional hash tables which stores the hash values of the keys after mapping them with the hash functions.Storing and retrieving the data when required from live overlay nodes is basic fundamental of this process.The KBR layer links the values from the overlay nodes to the keys.

The DOLR (Decentralized Object Location and Routing) as the name suggest gives a decentralized directory service.Here, each end point are represented by an object ID and they are positioned randomly in the system. The positions or the locations of the endpoints are declared by the applications itself.The ends points play an important role in delivery of the client message as the unique object ID is addressed on the client message and it is delivered to the nearest endpoint having its name.Annotating trees which are linked with object IDs can also lead to construction of distributed directory.Though, implementations by other methods can also be done.

The CAST, it allows group communication or intercommunication and synchronization among the nodes in the overlay system.Here, a single node does two processes generally ,either join or leave the group.Group representation is done in the form of tree and is decentrally managed.CAST,can support or handle large groups and multicast is effective and group member neighbours to anycast originator is used as a message delivery point ,if the overlay fully supporting Key Based Routing services is *proximity aware*.

4 Key Based Routing API

This section includes the basic techniques and methods which are purposed for Key Based Routng API which provides data based on interest of the host. Methodology, involved in API is described with help of data types .This section also focuses on the implementation of the DHT, DOLR and CAST in higher abstractions levels .

4.1 Data Types

The key consists an string of 160 bit.It has nodehandles which has the metadata of the nodes.The metadata which nodehandles contain generally comprises of IP address of nodes and port.Application data of various length is holded by the message *typemsg*.API is described by the usage of an language neutral notation.The parameter *p* is represented in two ways $\rightarrow$ p in case its only an read only parameter and in case of read write parameter it is represented as $\leftrightarrow$ p.T[]p is representation of an ordered set of parameter *p*.

4.1.1 Routing Messages

void route(key:K, msg:M, nodehandle:hint)

In this function the message is to be delivered to the root key.The nodehandle uses hint argument which declares the path or hop for routing the message.An hint argument is referred to current key's root which results in delivering of the message and is known as a good hint.Whereas, a bad hint can lead to multiple hop's.

void forward(key:K, msg:M, nodehandle: nextHopNode)

In this function message passing is done with help of upcalls which are invoked at each node.The forwarding of messages evoles the source node and the key's root node.The application is informed by the upcalls about message being forwarded by using key to nextHopeNode.For termination of message forwarding the nextHopeNode is set to be Null.

void deliver(key:K, msg:M) When message *M* is delivered to the node which also acts the root for the key K ,this function comes into operation.Upcalls are used to deliver the message which is also reliable and convenient for the applications.

4.2 Routing State Access

The routing state of the nodes is only accessed by the upcalls and there is no communication between the nodes.Next hop destination is decided on basis of routing state queries done by the application for gaining nodes.This step is done in order to forward the messages to nodes by the forward upcalls.

nodehandle[] local lookup(key:K, int: num, boolean: safe)

In this function a set of nodes are introduced which can be used as path or next hop for a key.The route is set in such a manner that it satisfies the overlay protocol's for the usage of hops according to the boundations or rules.

nodehandle [] neighborSet (int:num)

As name itself suggest this function creates a set of nodehandles which are nearby or neighbours of the local node in the identifier space.

nodehandle [] replicaSet (key:k, int:max rank) The functionality of this operation is to turn back the ordered set of nodehandles which contains the replicas values of the object along with key K could be holded.Nodes are returned by the calls having ability for maximum rank and rank up to.Maximum replica set is returned back in case the maximum rank goes beyond its criteria of max replica set size.There are some protocols which supports only one as their maximum rank value.In case of the local node failure ,the protocols which supports the rank value more than one , these returned nodes can be used for replicating data as they are likely to become the next root of the key k.

update(nodehandle:n, bool:joined)

This function involves passing of information to the application by upcalls for either joining or leaving of nodes from the neighbour set.Neighbour set calls are used for returning of the set.

bold boolean range (nodehandle :N, rank :, key :lkey, key: rkey)

It involves the information of the keys ,particularly their range for which node N is the root.The function is to be False if the range of the node could not be found out and in-case ,it is found out then it said to be True.

4.2.1 Clickstream data

In this part concept of data analyzing and obtaining from OSN sites is described with help of Social Network Aggregator.

4.2.2 Social network aggregator

It acts as an intermediate between the user and the online social sites.It's functionality is to fetch the data from the different networks into a single point which helps the users to manage their profiles easily though belonging to different networks.A user is able to access to all the OSN sites when the user login's via Social Network Aggregator.The user doesn't requires to login separately in each OSN site for the access instead the Social Network Aggregator provides a common interface to the user for all OSN sites. This is process is done by usage of two-level real-time HTTP connectionwhich is divided into two parts, the*First level* an interface between the user and social network aggregator and the *second level* interface is intermediate between the social network aggregator and the OSN sites.Open APIs provided by the OSN sites are generally used for intercommunication between Social network Aggregators and the OSN sites.The figure 4 below explains the functioning of the Social Network Aggregator.

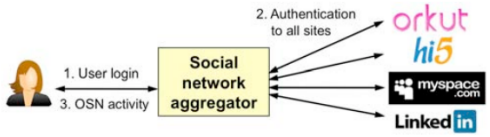


Figure 2. Illustration of a user connecting to multiple OSNs through social network aggregator [1]

All the data from OSN sites are provided to the user via common interface and also the Social Network Aggregator doesn't restricts any boundations for the OSN sites ,thus the users can access all the features of the OSN sites simultaneously.

5 Overview on Geographical Overlays

Geographical Overlays works on the principle of Location Based Search.This technology is widely used by most of the search engines and navigation systems.Every peer in the geographical overlays represents an information about an object.Updating and transmission of data regarding that object is done directly without keeping server as their intermediate, which also reduces chances of failure.The basic idea of geographical overlays is to obtain the geographical position or location of a peer and manipulate the data provided by the peer for further analyses or is simply given to another peer on the network.This technology includes examples like showing of Restaurants ,Petrol Stations,Police Stations etc on the navigation device of a car using GPS(Global Positioning System).This type of search is done by using two types of overlays and they are Globase.KOM and Geodemlia.

5.1 Globase.KOM geographical Peer-to-Peer Overlay Architecture

Globase.KOM is a peer to peer superpeer based overlay which is represented in form of tree and peers which are in zone of superpeer are interconnected with the it.The figure below shows that the earth is projected in form of rectangle and different zones which don't overlap with each other.Complete isolation is done regarding the boundaries of the zones.One superpeer represents as the head of every zone in which it belongs too,which also maintains the list of contacts of the peers in its zone.All the superpeers are represented in form of tree and **superpeer A** is known as the parent of peer**B** only when peer **B** resides in the zone of A *superpeer*.

According to [4] Each superpeer maintains the contact addresses of:

1. all the peers residing in its zone.Inner zones are excluded i.e child nodes linked to parent superpeers.
2. all the superpeers representing inner zones (i.e. Superpeer connected with child nodes).
3. contacts of parent in the tree structure.
4. root of the superpeer in the tree.
5. interconnections among the superpeers.

Every peer in the Globase.KOM maintains the contact addresses of the parent superpeer in the tree.It also maintains the contacts for the roots of the superpeer.Interconnections list among the peers in the tree is also maintained by the peers along with currently, contacted peers described in the cache list.

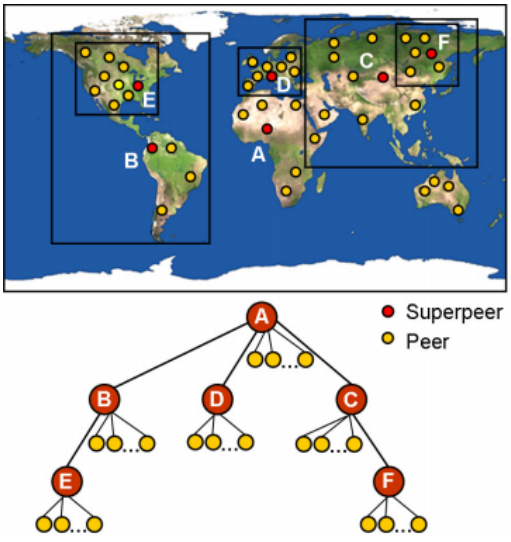


Figure 3. Design of Globase.KOM

All the peers have Unique ID and are identified by them only.The figure below describes the concept. The components of peer ID are :

1. Node's GPS coordinates
2. If it is a super node the zone of responsibility ,as shown in figure.
3. Random Part, for supporting of presence of multiple peers in same location

Globase.KOM is represented and described by *concatenation its vector representation*[4] and by usage of its side lengths and point at last of left side of the zone.

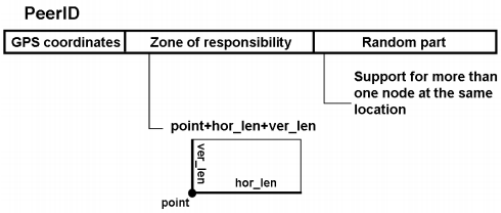


Figure 4. Structure of Peer ID

5.1.1 Look Up

This operation is used to find the IP address and port of the peer in its geographical location. Superpeers have all the knowledge and information example :ID's,Port etc of the nodes connected to it.A LOOK UP operation is basically in which a peer sends a message like LOOK UP to its superpeer and this message contains details like address and the sequence number of the peer.Later on,the superpeers finds out the location of that particular peer and checks that the peer is inside its zone or not.The superpeer forwards the request to child if it finds that the particular peer is inside of its zone. Look up messages are sent directly to the peer.If it is find out that the peer is inside the zone of interconnections of the superpeer.Alternatively, the message is sent to the parent superpeer of that node.The process is repeated by the parent superpeer.

5.1.2 Area Search

This operation is based on data of the geographical area of the peer.It works by usage of a Search message.According to [4] "Calculations of searched eclipse on to the map projections start as soon as the superpeers recieves a SEARCH query from its peer".The further step includes the investigation of the eclipse intersection with its zone.However,the functionality is similar to the LOOK UP ,but still it has difference like a SEARCH RESULT message is forwarded to the all the peers by their respective superpeers in the

searched area containing the list of matched peers.Time constraints are forced on the search and after the time interval the search is timeout.Usually, it takes 2 seconds for time out for the search.

5.1.3 Find the Closest Peer

While searching for the closest peer in geographical area ,a peer checks for the nearest border of its zone.This functionality works by usage of details of the parent superpeer such as its ID which contains the zone representation in vector form.Further more, this operation starts by sending a message FIND CLOSEST to its parent superpeer.The FIND CLOSEST message contains details of closest border of zone to the superpeer. Superpeer analysis the data and checks whether there is any peer in the same radius.If, there is any peer nearer to initial one then it is included in FIND CLOSEST RESULT message forwarded to peer. Incase, the superpeer is not able to find any result then it sends back FIND CLOSEST NEXT back to the peer which holds the address of the parent superpeer.Further, repetition of the steps takes place till the FIND CLOSEST RESULT message arrives to the peer holding the contact details of the neighbor peers.

5.1.4 Join

For joining process of a peer the bootstrapping superpeer generates a message called NEW PEER message in order to have the details of the zone in which the peer is placed.New peer ID is formed, and that peer is added to peer list by the new superpeer found. A message is sent back with PEER OK containing details like Peer's ID new one, interconnections and contact of superpeer root.

5.1.5 Leave

Parent Superpeer is informed by message REMOVE SUPERPEER while leaving the network. The message includes details like list of the peers,interconnections and child superpeers.After recieving the message the superpeer discards the child superpeer from its lists.The functionality of the peers and superpeers from the leaving child superpeer is then passed on the parent superpeer.After completion of removal process a new message REFRESH SUPERPEER is passed on the remaining child peers.

5.2 Failure in overlay

Incase,of a failure of a peer or superpeer in Globase.KOM it is found out when particular peer misses the 3 KEEP ALIVE message that too in a sequence.The failure is detected by the superpeer of the peers and superpeer generally discards the peers from its contact list in the overlay systems.

5.3 Failure of Superpeer

A KEEP ALIVE message is sent after fixed interval of time to the parent as well as child supepeers and further this message is passed to their peers.Incase, of failure of a superpeer all the peers in its list will not receive the KEEP ALIVE message.Therefore, the superpeer will send a KILL message to all the superpeers and parent superpeers which are failing to respond back.In the case of failure of a superpeer all the peers and the superpeers linked to it starts to find their positions , it is done in order to find out that they reside in the zone of their interconnections.If they reside in the same zone then they simply send TAKE ME message to connected superpeer , alternatively the TAKE ME message is sent to the root of the superpeer.The message is forwarded to the smallest zone in which the sensed peer resides.The receiver of the TAKE ME message is generally the parent of the superpeer which failed currently and if the parent superpeer is working then it will take the peer to its peer list.It can also make the sender superpeer as new child superpeer in the Globase.KOM and in any previous cases the parent superpeer forwards a REFRESH SUPERPEER to them.In a special case, when no peer is live the superpeer generally rejoins.

5.4 Failure of Root Superpeer

Particularly, in this situation one of the child superpeer overtakes the failed root superpeer to avoid a situation of several independent trees. *Election on Bully Algorithm is used*[4]. Interconnections are maintained among the child superpeers .Root superpeer uses a message REFRESH BROTHERS for refreshing of the contacts.When a failure of root superpeer takes place election algorithm is used by child superpeer for elections in order to choose a peer with highest ID.A election message is passed with a sequence number.After completion of the election zone of the root superpeer is taken over by the elected superpeer.A message REFRESH ROOT message is forwarded to all the peers in the network.

5.5 Overview of Geodemila P2P overlay

In present peer to peer overlays supporting location based search are reliable for client server architecture.One of the major drawbacks of this peer to peer approach is caused by joining and leaving of the peers leading to high churn.Geodemila is a new location based peer to peer overlay to gain a robust ,scalable and effective location based search.The design of Geodemila is relevant to the previous Kademlia Overlay.Geodemlia is basically designed for static peers interconnected through internet.It supports area search in terms of location based data and storage generally updated by mobiles or cellular phones.It is a key feature in peer to peer backend for area search on behalf of location based services.In Geodemila peers are allowed to generate search requests for location based data.Example

search petrol stations nearby on the highway in an radius of 1km from the current location.

5.6 Model of Geodemila P2P overlay

The functionality of Geodemila is basically on assumptions like unique position of the peer on the locations assumed to be sphere."The basic idea about using a spherical model is that localization techniques return spherical coordinates,which can be directly handled by Geodemlia[GSH 12]".Here,every peer can find out its location *lp*=(ϕ,ψ) by usage of localization methods.The technique or system which favors for location based search for a peer are Global Positioning system (GPS),IP locator etc.The longitude of peer is defined by using $\phi\ \varepsilon\ [-180°,\ 180°]$ and latitude is denoted by $\psi\ \varepsilon[-90°$,$90°]$.*For calculating the distance d(l1, l2)between two arbitrary points l1 =(ϕ1,ψ) and l2= (ϕ2,ψ)on the sphere.The haversine Formula shown in Equation below is used* [GSH 12]

$$d(l_1, l_2) = d(\phi_1, \psi_1, \phi_2, \psi_2) = 2r \cdot arcsin\left(\sqrt{a(\phi_1, \phi_2) + b(\phi_1, \phi_2)a(\psi_1, \psi_2)}\right)$$

with

$$a(\phi_1, \phi_2) = sin^2\left(\frac{\phi_2 - \phi_1}{2}\right)$$

and

$$b(\phi_1, \phi_2) = cos\,(\phi_1)\,cos\,(\phi_2)$$

Peers in the overlay are connected with TCP/IP so that they can intercommunicate with each other till they holds IP and port of each other.Data objects are linked to the geographical location and data is stored to support search engines for their information.The search in Geodemila is assumed on spherical shape .It is also possible that it can have other shapes example rectangular,square etc. Finally,each peer p and data object o is assumed to have a permanent identification position *lo*.Every peer *p* and data object *o* is *randomly identified by i ε[2,2^{160-1}]* [3].

5.7 FIND NODES(ls, k, b)

FIND_NODES(ls, k, b),for this suppose a request is generated on any random position like ls =(ϕ s,ψ s) to a peer *p* which receives the request and k is an integer value.The peer p is supposed to reply with k nearest peer.The responding peer must have knowledge and respond in accordance with the queried location *ls*.To avoid replication of the already found peers ,a bloom filter b which has a size 160 bits is used.This bloom filter b is attached with the request sent to receiver by the peer p.This concept helps the receiver peer P to have the knowledge of all the peers which have been received by the queried peer p,thus the peer P only submits the non-received peers to its response.

5.8 STORE(o, lo) r

STORE(o, lo)is an request regarding object o with location lo.In this function, the object is stored in such a manner that in conditions of high churn and also in the case of rapid failure of the peer ,the data of the peer should not be lost or damaged.*The overlay should store the object persistently*[3].A lifetime is specified by a peer for storage of data object for avoiding and duplication of outdated data and completion of the lifetime the data is finished or discarded.

5.9 AREA_SEARCH(ls, r, s, b)

AREA_SEARCH(ls, r, s, b) is a operation in which the dimensions of the peer of interest is denoted by using : ls =(ϕs,ψs) for longitude and latitude is found by using the raduis r and search term s.The system should support the condition d(ls, lo) ¡ r and the system should response at point lo =(ϕo,ψo) which matches the search term *s*.The term *s* represents to the objects such as petrol stations ,police stations etc in which might be useful for the user.Example: Locations of petrol stations , Restaurants etc are displayed on navigation system while the car is moving on a highway.Bloom filter is used to avoid already found data objects ,for this we use their IDs. Peers which already have area search request to them will not add the current information into their response ,avoid duplication of the data.

References

[1] Fabrcio Benevenuto, Tiago Rodrigues, Meeyoung Cha, and Virglio Almeida. Characterizing user behavior in online social networks. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 49–62. ACM, 2009.

[2] Frank Dabek, Ben Zhao, Peter Druschel, John Kubiatowicz, and Ion Stoica. Towards a common api for structured peer-to-peer overlays. In *Peer-to-Peer Systems II*, pages 33–44. Springer, 2003.

[3] Christian Gross, Dominik Stingl, Bjorn Richerzhagen, Andreas Hemel, Ralf Steinmetz, and David Hausheer. Geodemlia: A robust peer-to-peer overlay supporting location-based search. In *Peer-to-Peer Computing (P2P), 2012 IEEE 12th International Conference on*, pages 25–36. IEEE, 2012.

[4] Aleksandra Kovacevic, Nicolas Liebau, and Ralf Steinmetz. Globase. kom-a p2p overlay for fully retrievable location-based search. In *Peer-to-Peer Computing, 2007. P2P 2007. Seventh IEEE International Conference on*, pages 87–96. IEEE, 2007.

Overlay Evaluation: Testing and large-scale tests

Bing Gao
Project Group - A Peer-to-Peer Framework for Social Networks
Universität Paderborn, Supervisor: Dr.-Ing. Kalman Graffi
Email: binggao@mail.upb.de

Abstract

In Internet-based P2P networks, communication architectures are handled with a large scale systems where nodes frequently join and leave. How evaluated network overlays are vital for quality of system? This paper discusses about the challenges which are being faced in testing large-scale systems. Then discuss testing phases, approaches testing methods and some architectures which would be beneficial to large-scale systems with volatility. After the discussion of these approaches, we get a approach which would be appropriate for testing our Peer-to-Peer system. Specifically we discuss further work and challenges we are facing.

1 Introduction

A huge number of people are using Internet. Social media is around us almost everyday. And the social media is applied to different devices like Smarter phones and tablets. This makes social network more and more popular. In recent years, Peer-to-Peer (P2P) systems quickly became a popular topic in data sharing area. In P2P systems, the participants share resource to other nodes. They are clients and at the same time they also act as servers. The advantages of P2P systems like high function/price ratio, robustness promote developers to do exploitation in this area. Comparing to traditional centralized communication systems, large systems like P2P-based communication system can exploit the heterogeneity of the devices, thus the problems like disaster management and effectiveness of communication can be solved [2]. According to statistics, over 50% of the development costs are used for testing in software development process [18]. The importance of testing in software quality assurance results in the attempt of finding the effective and appropriate testing architectures. The testing work for P2P systems has turned to be challenging because the systems must be deployed on a large scale of nodes, and the volatility of the large number of nodes makes the testing work more tougher to proceed. Benchmarking of Large-scale Systems is complex, not only because the scale is large, the large number of components in the system is also a reason. Additionally, the toolsets we could obtain are limited, sometimes we have to write our own test grips for benchmarking because we may not get any tool to specify our own benchmarks. So it's a kind of black art, for example some tester has done much benchmarking and testing work based on his skills and knowledge, when it comes to another person to do continue the work, the efficiency of benchmarking and testing is not on par with his skill. So there should be a transfer of knowledge and skills to the next person. Thus we are trying to find a methodology to quantify the transfer of skills.

And due to the volatility of the system, it may be considered as a fault during tests if some nodes reject to answer the requests from other nodes and even leave the system unexpectedly. So we have to present a testing framework which allows test cases to control the volatility of nodes [8].

This paper first introduces some basic concepts and requirements of testing methodology of P2P systems. Then introduces the three phases for testing a large-scale system. In the next section "Testing methods for large Scale systems", we first discuss about the preparation work before which in necessary for testing a system, then talk about test methods from the point of functionality and performance evaluation for large scale systems, and we will analyse the main difficulties with volatility testing in a large system. In the section of "Testing Architectures for Large Scale systems", we discuss about some approaches which are designed base on the typical centralised testing method of distributed system from the point of synchronizing execution of the actions in test cases, and the main problems they can solve. We especially expound the advantages of these approaches which the author considers them as better solutions specially for large-scale systems like Peer-to-Peer systems. These two approaches are "B-tree architecture" and "Gossiping architecture". Then we will analyse these approaches through a test case sample. In the section "Related Work", some useful tools which play important part during the testing phases are enumerated. In the last section of "Conclusion", we discuss about the challenges of P2P

testing work and what we intend to do as future work.

2 Basic Concepts

The objective of testing framework is to verify if the features of a system are working correctly according to the expected quality criteria: robustness, correctness, completeness, performance, security, etc [8]. We execute test cases to deduce if the behaviour of system conformance to our requirements. A test case is composed of the name of it, an intent, the input data and the output according to our expectation.

A test suite is the main class of the testing application, it contains several test cases.

A test case $\tau = (A^\tau, T^\tau, L^\tau, S^\tau, V^\tau)$. where $A^\tau \subseteq A$ is an ordered set of actions $\{a_0, ..., a_m\}$, $T^\tau \subseteq T$ is a set of testers which will execute the actions on the same logical nodes of the system, L^τ is a set of local verdicts, S^τ is a schedule,V^τ is a set of variables. [6].

Schedule A schedule is a map from A to T.

Action is a tuple $a_i^\tau = (\Psi, \iota, T^{'})$. where $T^{'}) \subseteq T$ is a set of testers that executed this action. Ψ is a set of instructions, ι is the interval of time in which Ψ should start being executed.

Local verdict is the result from comparing the execution result R with the expected result E. The local verdict ν of test case τ on the interval of time ι is defined as l_ι^τ :

$$\begin{cases} pass \ \ if R = E \\ fail \ \ if R \neq E \\ inconclusive \ \ if R = \phi \end{cases} \quad (1)$$

[6]

Each test case involves some certain set of actions which will be executed by Distributed Hash Table (DHT) on the set of peers representing the P2P systems. DHT maintains the mapping from a range of keys to the corresponding storage of value[7]. Test suites which are deployed on some logical nodes constitute a tester. Typically one node from the system acts as the coordinator or schedule all the test actions during testing. The tester and the coordinator are the main components in a testing framework. The testers are deployed by some nodes of the system to observe and manage test case actions which are implemented as annotated methods. And these annotations are very helpful because from them we can get the description of behaviours of the test case actions. With annotations a tester can get the actions which are annotated to it and manage the actions. The coordinator is used to synchronize execution of test cases among the testers. All the testers register their actions with the coordinator. Then the coordinator builds a schedule with which it can coordinates all the actions to be executed.

Figure 1 shows the process of schedule construction and test case execution.

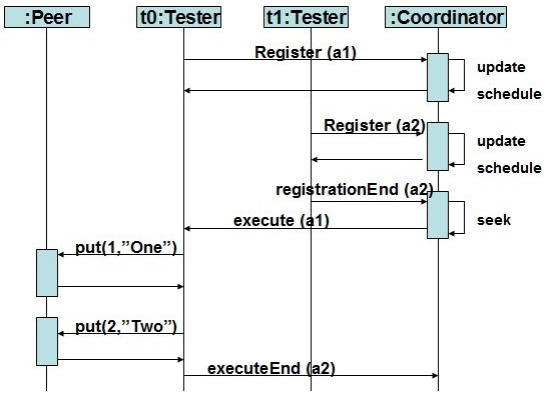


Figure 1. Process of scheduling construction and test case execution

Here we assume two actions (a_1, a_2) will be executed on this single peer. a_1 is to insert the string "One" to key 1; a_2 is to insert the string "Two" to key 2. The coordinator asks for a local verdict from the testers after finishing all the executions of these actions. After receiving the number of pass verdicts and fail verdicts, we get the testing result. Comparing with the expected ratio of pass and fail, we can know if the verdict is pass. A testing framework can be implemented by some programming language (Java,C etc.) or by a testing framework tools like JUnit.

3 Phases for Large Scale Systems Testing

There are three phases for Large Scale Systems Testing. α testing, β testing and γ testing.

3.1 α Testing

α testing is to organize the developers to simulate or actual operational testing the system from developing side, thus find out the errors and eliminate them in developing environment. The purpose of α testing is to evaluate the acceptance, functionality, usability, reliability, performance and support. The key feature of α is to simulate realistically actual operating environment and operations from various users. When α testing reaches a certain level of reliability, only β testing can begin.[17]

3.2 β Testing

β testing is performed by real users of the system in the actual user environment for which can not be controlled by developers. These users give feedback about the error information to the developers. Developers are not in the field during β testing. Most of imperfections of the system are eliminates after β testing. It can be considered as another form of user acceptance testing. And the system turn to be quite mature, but some flaws or loopholes may still remain.[17]

3.3 γ Testing

In the phase of γ testing, only some optimization is needed. Generally, γ testing is rarely used because almost all the defects are fixed during α and β testing phases. So during γ phase, we test again to find whether there is still defects in the system or any functionality can be improved. Then the software can be published.

In these three testing phases, α testing plays a very important role because it can simultaneously going on with the developing process. It can be controlled by developers in the developing environment. It can start right after the development of some sub-module of the system gets done. The feedback from α testing helps developers to find and solve problems easy from basic levels, then proceed with design and development. So for α testing, some testing methods and architectures are discuss in the following sections.

4 Testing methods for large scale systems

In this section, testing methods are discussed. Before really start to test the system, some preparation work is still needed to get down. Based on the requirements of testing a system, functionality, performance and volatility of the system should be tested.

4.1 Preparation for Testing a System

First let us see some important preparation for testing a system:

- Must know the components well and how they are setted up and configured, it is very important because the setted up and configuration directly impacts the expected performance, and a component's performance response may be different if it is exercised differently.
- A clear understanding of the critical data path has to be established.
- The testing should be repeated for verifying the variability. Because an obtained performance may vary with time even under the same test conditions.
- Learn the reporting and statistics mechanisms of the system. Because the expected result and the result we get from test might show disparity, that is a way to help us to understand how the system is operating.
- Test from bottom-to-top, we should start starting from the very basics and from the bottom of the backed system. This helps to get better understanding how the errors or disparities happen.
- Having an expected result before actually running any test,this will help analysing the results and tester's understanding of the underlying hardware.

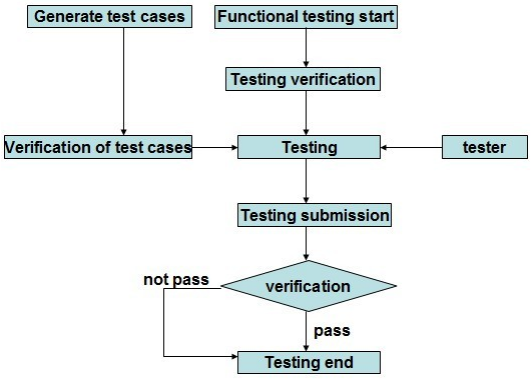


Figure 2. flow chart of the functional testing

- Right selection of tools based on the hardware and components of the system.
- Good recording work and documentation are important for the test result.

This is the preparation work we should do before starting testing a system. After the preparation gets done, we should design a process for the testing work. A clear process benefits effective testing work. Because there are many sub-modules in a full-featured large-scale system, it's definitely necessary to evaluate the functionality, reliability efficiency, etc. Below are processes of functionality testing and performance testing:

4.2 Functionality Testing

The purpose of functional testing is to verify whether the system has the requisite functionalities.

Figure 2 shows the flow chart of the functional testing as follow[20]:

Here the test cases are composed of test case ID, test case name, test case objective, expected result and so on.

And to reduce resource consumption during the testing of the system performance , we can use virtualization technology in which we only need to handle the testing work with a small number of physical computers [13]. And for handling with detected errors, testing a single function of the system in a random or round-robin order decays error rates. So from the point of functionality, we can use some linear software reliability models to estimate the number of errors generated from the testing [21].

4.3 Performance Testing

The purpose of performance testing is to verify whether the performance of the system achieves the requisite performance targets. Figure 3 below is the flow chart of performance testing[20]:

The testing tools in this process are simply classified as pressure loading testing tool(LoadRunner, WebRun-

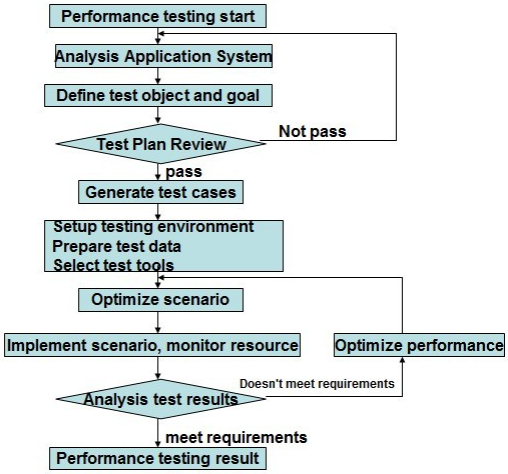


Figure 3. flow chat of performance testing

ner,etc.), resources monitoring tool, fault location tools and optimization tool[20].

4.4 Volatility Testing

Based on the requirements for testing a system : functionality, scalability and volatility [16]. Testing work turns to be tough when we consider the volatility of a large scale distributed system. Now let us analyse volatility step by step. Typically when testing the volatility of a system separately, functionality doesn't take account. So the testing environment can be divided into four parts:

- small number of peers without volatility.
- small number of peers with volatility.
- large number of peers without volatility.
- large number of peers with volatility. [8]

In the first case, the system is stable and with countable number of peers. Other peers can retrieve the variation when any peer of them insert some data into a DHT, because every peer knows others properly. It's even possible for the peers to exchange data directly with each other. This case can be expanded to the situation of large scale stable systems if these test sequences are parametrized by the number of peers. In the second case, we only need to concern about the pure functional problems: data forwarding and routing table update. When peers join the system after a peer inserted data, the data should be forwarded and the routing table must be updated on time. For large scale systems without volatility, based on the scalability, what we need to do is enabling the peers to perform some tasks like routing messages and forwarding data to other peers. Now we see the fourth case since it is for Large-scale systems with

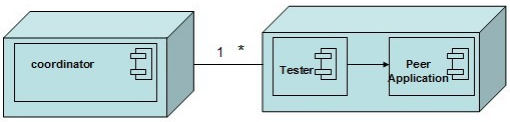


Figure 4. deployment diagram of this testing framework

volatility, the problems for testing this kind of systems are related to three dimensions: functionality, scalability and volatility. It is a very complex process while all these three aspects should be taken into account [8]. Compare to the situation of every node knows other ones in a small scale system, functionality performs obviously different because every peer only has a partial view of the whole system. At the same time peers are leaving and joining, every time after a peer leaves or joins, all other peers have to update their routing table. And we have to build a testing framework based on such environment.

In the framework for testing the volatility of a large-scale-system, there are two components: the coordinator and the testers. The coordinator is an application that dispatches test case actions to testers. The testers are the applications execute on some nodes, control volatility according to the requirement of test cases. Fig.4 shows the deployment diagram of this testing framework.

Figure 4 above is the deployment diagram of this testing framework:

Here the coordinator dispatches the actions of the test case A^τ to the testers in T^τ. The testers are responsible to execute the actions and control the volatility of the nodes.

When volatility of the system changes, the coordinator makes peers leave or join the system, the routing table has to be uploaded correctly and the requests have to be routed. So the coordinator has to update its schedule every time correctly when the volatility of nodes changes, then only we get the local verdict $pass$. If the coordinator is not informed when a node left the system, the testing can not be proceed because it is unable to follow the sequence, then we get the local verdict $fail$. If errors occur during a test procedure and the errors may be related to the changes in external specification of the system, then we may get the local verdict $inconclusive$, in this case we can design a procedure to identify the effective test regions for detecting incorrect volatility [15].

While the coordinator is also a node in the system, as we discussed above, the coordinator coordinates all the actions to be executed by a schedule. So actually the actions in test cases are decomposed by the centralized node (the

coordinator), then dispatch them to different nodes. And these kind of centralized testing methods are the typical distributed testing architecture for testing the volatility of a large-scale-system. Some sets of machines like JUnit, FIONA, Joshua, GridUnit are used to distribute the execution of test cases.

Based on this approach, the test case is executed by some of the testers. But we also need to know by which way each action is dispatched to the assigned tester, because the organization of testers concerns the control of a test case. That is one problem. And another problem, some machines for distribute the execution of test cases like Joshua, different nodes can run different test cases, but a single action in the test case is executed only once on some node as the coordinator dispatched. We don't get verdict from other nodes which also have the ability to execute the same action. Nodes send local verdicts to the coordinator after the execution of the test case, except this, there is no more communication between them. The relation of coordinator and testers is unidirectional: testers execute the actions as the coordinator dispatched and send local verdicts after executing the test case, and every action is only executed once on some node. That is the boundedness of this testing approach. For large scale distributed systems, the centralized node (coordinator) should has the ability to scale up all the nodes who participate in the testing. That's the scalability of a good testing method. Thus, based on this logic, we can build some testing architecture which can solve the testers organization problem. And architecture for solving the testing boundedness problem.

5 Testing Architectures for Large Scale Systems

As we discussed above, the objective of a testing method for large scale systems, is to ensure the correct execution of the test cases, at the same time the test cases should scale up all the nodes who participate in the testing. The previous section expounded this lack of requirement in the typical testing architectures.

Two solutions are proposed: "B-tree architecture" and "Gossiping architecture". They are originally exploited by Eduardo Cunha de Almeide, Gerson Sunye, and Patrick [9]. We believe that these two approaches are advisable, the features of them are discussed below. B-tree architecture gives a well-proportioned organization of testers. Gossiping architecture scales up all the nodes which participate in the test case.

5.1 B-Tree

The main idea of B-Tree architecture is about organizing the testers in a B-Tree structure. We use the coordinator as a controller of the system and the root of the tree. Each node in this structure has two children testers. For the actions

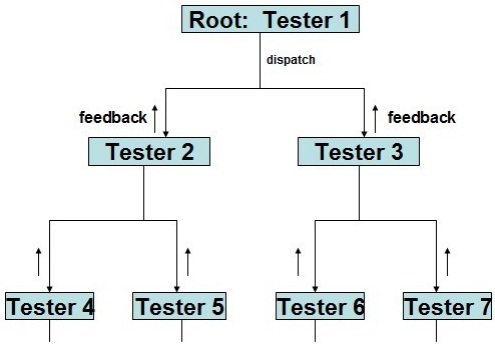


Figure 5. B-Tree architecture

in a test case, every parent node dispatches the actions to their child testers, and after executing the actions assigned to them, child testers reversely send the results to their parent testers, parent testers transmit the results to their own parents until the root receives all the results and generates a global verdict for the test case [9].

Figure 5 is an example of B-Tree architecture.

Tester 1 acts as the coordinator in this struction. It dispatch the test case to its children tester 2 and 3. Tester 2 and 3 execute the actions assigned to them, send local verdicts to tester 1 and dispatch other actions to their children tester 4, 5, 6, and 7. Tester 4, 5, 6, and 7 can not dispatch the actions because they are the leaf nodes, they execute the actions assigned to them and send local verdicts to the parent nodes tester 2 and 3. Tester 2 and 3 transmit the verdicts to the root. Then tester 1 gets all the local verdicts and generates a global verdict.

The goal of this architecture is to build a well-proportioned structure for managing testers. We have to build this tree before starting testing the system. But the order of testers is not fixed. It can be random.

5.2 Gossiping

Now let us search for some approaches with which the test cases can be executed on all the testers even the entire system. After of execution of the test case, let the actions be executed on every node which has the ability to execute some actions. The rest is similar with B-tree. After execution each node sends a local verdict for every action to a central node who acts as the coordinator in the system. Then the coordinator combines the verdicts and gives the global verdict of the test case.

In order to make all the nodes currently involved participate into the test case, we need to give abilities to the testers, make them communicate with the coordinator. This can be achieved by using traces produced by each node. Each action can be executed on different testers to which the action is assigned , the testers give local verdicts to the controller which will generate a global verdict for the test case. This

Action	Nodes	Instructions
(a0)	0,1,2	Join the system;
(a1)	2	Insert “one” at key1; Insert ”two” at key2;
(a2)	*	Pause;
(a3)	0	Retrieve data at kay1; Retrieve data at kay2;
(a4)	1	Leave the system;
(a5)	0	Retrieve data at kay1; Retrieve data at kay2;
(a6)	0,2	Leave the system;
(v0)	0	Calculate a verdict;

Figure 6. the ordered actions managed by the testers

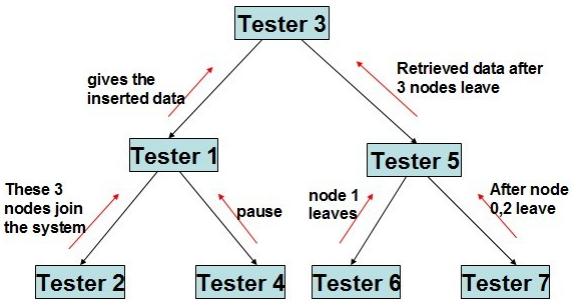


Figure 7. Testers managed B-Tree architecture

communication between the coordinator and the testers in the systems has to be bidirectional, scalable and fast. That is the key feature of this architecture for a large-scale-system.

Gossiping is also a centralized architecture like B-Tree. The differences are, the actions are executed by gossiping the coordination messages among the testers, it's an bidirectional communication between coordinator and tester, and the actions should be executed by a schedule which contains a execution sequence. The coordinator contains this schedule and sets a multicast address for every test case action. Then the test case is stored at each tester in the system. Each tester will recognize the actions it should execute and subscribes to the multicast addresses of these actions. At the end the first action starts being executed on the testers assigned to them. And other actions follow on by sequence.

Compared to B-Tree architecture, this approach has a improved feature: it guarantees the execution sequence of the test case actions. And the actions can be executed many times on different testers. This improves the accuracy of the test [9].

5.3 Test Case Sample

Now we implement a test case sample to see how do these two architecture work and to compare their performance.

Here we assume that on three nodes p_0, p_1, p_2, an action set $a_0, ..., a_6$ is managed by a set of testers: t_0, t_1, t_2.

Figure 6 shows the ordered actions managed by the testers:

In the tester set, t_0 is responsible for giving local verdict. First, we populate the DHT with actions a_0,a_1 and a_2. After p_1 leaves the system, if the data at key 1 and key 2 can still be retrieved and they are the same as the data inserted by p_2, then there is no error on DHT implementation in this test case. Otherwise the verdict is fail or inconclusive.

5.4 Solution Based on B-Tree

Now let's try to solve the test case sample by using B-Tree architecture. Here we need to increase the size of tester set appropriately. The number of testers which are included in testing may vary according to the number of actions in a test case. We manage the testers in a B-Tree structure by which the actions are dispatched. Then the way to dispatch the above test case sample can be described as Figure 7.

From Figure 7, we can see here tester 3 is the root in the B-Tree. Tester 3 dispatches actions to tester 1 and tester 5, tester 1 dispatches actions to tester 0 and tester 2, tester 5 dispatches actions to tester 4 and tester 6. tester 0, 2, 4 and 6 do not dispatch any action.According to the above test sample, here a_0 is executed by tester 0, a_2 is executed by tester 2, they give results to tester 1. Then tester 1 starts executing a_1. In the same way, a_4 is executed by tester 4, a_6 is executed by tester 6, they give results to tester 5. Then tester 5 starts executing a_5. Tester 1 and tester 5 give results to the root tester 3. After getting the results for their children, Tester 3 compares the original inserted data and the retrieved data after the 3 nodes leave the system, get the test case result.

5.5 Solution Based on Gossiping

Here we give a sequence by which the actions should be executed in our test case sample: $a_0, a_1, a_2, a_3, a_4, a_5, a_6$. Initially any node can be chosen to be tester 0 which acts as an identifier to all the other testers. The other nodes receive an identifier n from tester 0. Then t_0 creates a multicast address for each test case. The test case is stored at tester 0 and other testers. All the testers verify which actions they should execute and subscribes to the suitable multicast address. After this the testers should execute a_0 start the execution and send the local verdicts to t_0. Then other testers responsible for a_1 to a_6 follow by the action sequence. The testing process with a sequence can be controlled by the management control loop augmented by a feedback-based decision-support tool[19].

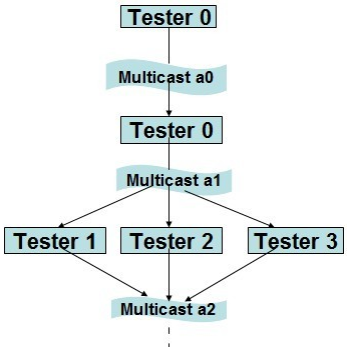


Figure 8. Testing managed by Gossiping architecture

Figure 8 shows the first action $a0$ being executed by testers $t0, t1, t2$. When the execution of $a0$ is done, the testers gossip the completion to the multicast address of a_1 and so on. Each tester sends it's verdict to t_0, then t_0 assigns a verdict for each test case.

6 Related Work

Before testing work starts, a detailed test plan has to be built, tools could be useful to the testing work should also be considered. Although the test plan may also be changed according to actual conditions in the testing process, the test plan it should be known whether it is good enough. This question can be answered by some evaluation criteria, and test plan heuristics can tell whether the test plan meets evaluation criteria [1].

In the testing process, the testers discover the weakness of system by following the tracks and evaluation results of test cases. Not only the testing architectures, the selection of evaluation models also plays a vital role for the overlay evaluation [14]. Their interaction influences on the software quality assurance. For measuring the processes with test data, the assessment methodology process-oriented metrics-based assessment (POMBA) can be used to improve the test cost-effectiveness [11]. This method makes the testing process effective and less resources can be wasted.

Two kinds of functional testing tools are very useful for the analysis of data transformation: data flow analysis tools and test data coverage tools [12]. These tools which play important part in the testing process can also be used when it comes to our development of P2P system.

For designing the testing environment, we can use D-Cloud computing technology which automatically configure fault injection along the testing scenario description [10]. And for addressing testing issues which are caused by immature testing appoaches which may result to testing without any clear goals, a testing maturity model (TMM) has to be developed [3]. This helps to solve many problems which happen in the testing process based on some testing architectures.

7 Conclusion

This paper introduced about some basic concepts of testing large-scale system, the phases of testing, testing methods, typical centralized approach for testing the volatility of a large-scale-system like P2P system. Based on this approach two architectures to organize the tester well and to solve the unidirectional problem between coordinator and testers are discussed. These two architectures are advisable because the bidirectional communication realized scalability of the testing. The only problems of them: a tree structure has to be built in the beginning of each execution in B-Tree architecture. And for Gossiping architecture, the executions of actions are consecutive and require $O(n)$ gossiping messages, that costs more than other approaches.

No testing method is perfect, this is no single method which can be used on testing the entire system, sometime developers have to make some testing method themselves for testing some specific aspect of the system, but again they have to break it cause it may be not able to satisfy other testing requirements [5]. Thus we have to select different approaches according to specific requirements.

When it comes to test a large-scale system like P2P social network, we may need to combine some of methods and approaches to satisfy different requirements of the system. As future work with the testing work for large scale system, we will try to get a solution considering the combination facts of volatility, scalability and functionality. And in practice we also need to find the way to improve the test-case effectiveness by making assumptions and understandable documentation, guarantee of the correctness of actions and so on [4].

References

[1] J. Bach and Principal. Test plan evaluation model. 1999.

[2] D. Bradler, J. Kangasharju, and M. Mhlhuser. Evaluation of Peer-to-Peer Overlays for First Response . 2007.

[3] I. Burnstein, T. Suwanassart, and R. Carlson. DEVELOPING A TESTING MATURITY MODEL FOR SOFIWARE TEST PROCESS EVALUATION AND IMPROVEMENT . 2007.

[4] Y. Chernak and V. F. Consulting. Validating and improving. 2007.

[5] J. L. DALLEY. The art of software testing. 2007.

[6] E. de Almeida, G. Sunye, Y. Le Traon, and P. Valduriez. Testing peers' volatility. In *Automated Software Engineering, 2008. ASE 2008. 23rd IEEE/ACM International Conference on*, 2008.

[7] E. de Almeida, G. Sunye, Y. Le Traon, and P. Valduriez. Testing peers' volatility. In *Automated Software Engineering, 2008. ASE 2008. 23rd IEEE/ACM International Conference on*, 2008.

[8] E. C. de Almeida, G. Suny, Y. L. Traon, and P. Valduriez. Testing peer-to-peer systems. 2009.

[9] E. C. de Almeida, G. Sunye, and P. Valduriez. Testing Architectures for Large Scale Systems. 2009.

[10] T. Hanawa, T. Banzai, H. Koizumi, R. Kanbayashi, T. Imada, and M. Sato. Large-scale software testing environment using cloud computing technology for. 2008.

[11] Y. He, H. Hecht, and R. Paul. Measuring and assessing software test processes using test data. In *High Assurance Systems Engineering, 2000, Fifth IEEE International Symposim on. HASE 2000*, 2000.

[12] W. E. HOWDEN. Functional Program Testing and Analysis . 2007.

[13] G. hun Kim, H. choun Moon, G.-P. Song, and S.-K. Shin. Software performance testing scheme using virtualization technology. 2008.

[14] A. Jin-Xia, W. Guo-Qing, L. Shu-Fang, and Z. Ji-Hong. Dynamic Evaluation Method Based Multi-Dimensional Test Coverage for Software Testing. 2008.

[15] T. Kaneko and T. Nakajo. Method of determining effective software testing regions:detection of software defects related to external specification changes. 2008.

[16] M. Lehn, T. Triebel, C. Gross, D. Stingl, K. Saller, W. Eelsberg, A. Kovacevic, and R. Steinmetz. Designing Benchmarks for P2P Systems. 2010.

[17] G. Liu, S. Huang, and X. Piao. Study on Requirement Testing Method Based on Alpha-Beta Cut-Off Procedure. In *Internet Computing in Science and Engineering, 2008. ICICSE '08. International Conference on*, 2008.

[18] J. Mayer. Message of the Program Chairs of STEV07. 2009.

[19] S. D. Miller, R. A. DeCarlo, and A. P.Mathur. Modeling and Control of the Incremental Software Test Process . 2007.

[20] Z. Qingfeng and Z. Qiangqiang. Testing method for large-scale software . 2007.

[21] M. Trachtenberg. The linear software reliability model and uniform testing. 2007.