Seminar P2P im Wintersemester 2011

Teilnehmer des Seminars im Wintersemester 2011

22. Mai 2012



Veranstalter:
Fachgebiet Peer-to-Peer Netzwerke und Fachgebiet Telekooperation
Fachbereich Informatik
TU Darmstadt

Inhaltsverzeichnis

1	von Heiko Dittrich	3
2	Virtual Coordinate Spaces von Beatrice Friess	10
3	Attack and Counter Measures in P2P based Streaming Systems $von\ Richard\ Gross$	18
4	Development on Friend-to-Friend networks von Jan-Michael Heller	28
5	A Survey: Monitoring and Measurement Techniques in Peer-to-Peer File-sharing Systems $von\ Gerold\ Hintz$	35
6	Semantisches Routing in Peer-to-Peer Systemen $von\ Benjamin\ Mohr$	44
7	Failover und Switchover von Diensten und Anwendungen in verteilten Systemen $von\ Sebastian\ Puttkammer$	51
8	Bestimmung des Anonymitätsgrads von Nadine Trüschler	57
9	Censorship Resistant Publishing Systems von Julian Wälde	65
10	P2P Service Delivery in Mobile Ad-Hoc Networks von Gregor Wicklein	70

Analysing the Security of Incentive Schemes in P2P-based File-sharing Systems

Heiko Dittrich dittrich@rbg.informatik.tu-darmstadt.de Department of Computer Science, Technische Universität Darmstadt

Abstract—For Peer-to-Peer (P2P) file-sharing services cooperation is essential. However, peers behave rationally and try to maximise their benefits, while minimising their effort. To increase system performance incentive schemes are used to enforce cooperation. Nonetheless, so called free-riders and malicious peers try to attack these incentive schemes. In this paper we will first explain two popular P2P file-sharing applications (BitTorrent and eMule), especially focusing on their incentive mechanisms. We then discuss incentive schemes in general, introducing genuine and artificial incentives and providing a taxonomy for incentives. We show that genuine incentive are regarded robust, since they are implemented directly into the service. Finally, we discuss attacks on P2P incentive schemes. We categorize the attacks in four classes: rational attacks, Denial-of-Service (DoS) attacks, ID-based attacks and collusion attacks. We show that with these attacks, it is possible to bypass incentive mechanisms and disturb P2P file-sharing services. We show that numerous attacks against BitTorrent are available and the current BitTorrent and BitTorrent private tracker incentives can be bypassed completely. We will further discover that eMule is insufficiently reasearched and few attacks exists. Finally, we introduce countermeasures, which prevent most of the analysed attacks.

Index Terms—peer-to-peer systems, file-sharing, incentives, free-riding, BitTorrent, eMule, rational attack, DoS attack, whitewashing, ID-mapping attack, Sybil attack, eclipse attack, collusion

I. Introduction

Peer-to-peer systems have been widely adopted in the internet today. There exists numerous P2P systems for several kinds of services. One of the most used services is file-sharing. Studies have shown that P2P file-sharing applications are responsible for more internet traffic than any other application [20] while P2P systems produce 60% of the overall traffic [5]. A study by Schulze et al. [32] from 2009 confirmed, that years after these first studies P2P traffic still is responsible for up to 69,95% of the internet traffic and that BitTorrent is the most used protocol. Through the scalability of P2P systems, a file-sharing application can easily provide vast amount of files at a relatively low effort for the participants in the network. Hereby a rational peer tries to minimise its effort while trying to maximises its benefit from the application. This behaviour is called free-riding. For example, a peer in a file-sharing application wants a high download speed and a low upload speed. If it is possible for peers to download without contributing to other peers this obviously decreases the overall system performance and can lead to system death. Adar et al. [1] and Hughes [13] have shown that in the Gnutella network the

majority of peers do not contribute at all to the system and that free-riding can be a serious problem which needs special treatment. Peers may have more sinister agendas and try to actively interfere with a P2P application or even try to render a P2P application unusable. These malicious peers perform attacks against the network structure. The intention of these attacks may span from planting polluted files (e.g., worms and viruses) to removing files from the network (e.g. for copyright enforcement). Modern P2P file-sharing applications implement incentive schemes to enforce cooperation (and reduce free-riding). Incentive schemes try to motivate peers to actively participate or enforce cooperation through the implementation of various algorithms like tit-for-tat and peer auditing. These incentive schemes are suspect to attacks by free-riders and malicious peers. However, a well designed incentive scheme is capable of resisting attacks, issued by free-riders and malicious peers.

The rest of the paper is organized as follow: In Section II we will explain two popular P2P file-sharing applications: BitTorrent and eMule, especially focusing on their incentive mechanisms. In Section III we provide an overview of the variety of incentive schemes and introduce an abstract classification. Finally, in Section IV we describe attacks against P2P systems and incentive schemes, especially focusing on BitTorrent and eMule, and discuss appropriate countermeasures.

II. APPLICATIONS

Two of the most used P2P file-sharing applications are BitTorrent and eMule. Both of these applications use a centralized server to allow peers to discover other peers. However, the design of these applications differ in many ways. A common feature of recent versions of both applications is the implementation of the *Kademlia* distributed hash table (DHT) [27]. However, the two systems are not compatible since they use different implementations of Kademlia [24].

Kademlia is implemented as a tree structure, with the nodes being the leaves of the tree. Each peer randomly picks a 160-bit node ID, determining its place in the tree. Each node is responsible for the key IDs, also having a 160-bit length, *closest* to its own node ID. To query a key, a peer simply has to search for the responsible node. Since Kademlia being an overlay network, meaning it is completely P2P-based and does not use a centralized server, it enables malicious peers to perform different attacks. A completely P2P-based DHT in

cooperation with a centralized server reduces the impact of a DoS attack against the server, whilst it allows malicious peers to start ID-based and collusion attacks.

In the Subsection II-A we will describe the BitTorrent protocol briefly and then analyse the build-in incentive. We will also shortly describe the concept of private trackers. In Subsection II-B the same is done for the eMule protocol.

A. BitTorrent

Cohen [6] describes BitTorrent as not solely based on P2P. For peer discovery a centralized server, called tracker, is used. These trackers are only responsible for a limited number of files. To download a file, a peer has to obtain a BitTorrent file (with the ending .torrent), this file contains all the information needed to connect to a tracker as well as meta information about the file. After he receives such a file, he connects to the tracker and receives a list of peers (usually 50) which have the complete file (so called seeders) or have parts of the file (so called *leechers*). It then connects randomly to a pre specified number of peers. All peers which are interested in a specific file form a swarm. So in reality the BitTorrent network consists of several, independent swarms. The introduction of Kademlia has added another way to discover peers. A peer may also query Kademlia with the hash-value (which is known from the .torrent-file) of the file to discover peers. Thus eliminating the need for a tracker and enables the peer to find peers which share the same file, but are connected to another tracker.

Unfortunately, some BitTorrent clients have their own implementation of Kademlia and therefore not all systems are compatible¹. BitTorrent does not generate an ID for communication with the tracker and other peers. The IP and port of a peer are used instead. A file is divided into multiple pieces which consist of multiple sub pieces. Each peer informs the other peers of the pieces it already has downloaded. A leecher may download a piece from another peer, requesting sub pieces until the requesting peer has the complete piece.

BitTorrent uses a built-in tit-for-tat incentive mechanism for data exchange to increase cooperation. For the piece exchange, a peer opens multiple connections to other peers bit *chockes* (singaling the other peer that he intents not to upload any data) the connection of all peers except a small number of peers (usually 4). To these peers the peer intends to upload pieces to. Every 30 seconds the peer performs an *optimistic unchoke* and randomly unchokes a peer. After 10 seconds it evaluates the upload and download rates and checks weather the unchocked peer provides a better download rate. If so, the connection to the peer with the lowest download rate is choked. Otherwise, the unchocked peer is chocked again. After a file has been completed, the strategy of the peer

After a file has been completed, the strategy of the peer changes. It then tries to maximises its upload rate instead of its

¹Clients which implement Kademlia and are compatible include the reference client (BitTorrent), μTorrent, Transmission, BitComet and Deluge. The first client to ever implement Kademlia, Vuze (formerly known as Azureus), is not compatible to the above mentioned clients.

download rate. However, effectively the algorithm remains the same. Another part of this incentive scheme is the *snubbing* process. If a peer does not send any sub pieces to another peer but receives sub pieces from that peer, the other peer may snub the unwilling peer, thus refusing to upload any pieces to the unwilling peers, until the peer receives sub pieces from the unwilling peer.

An additional incentive used in the context of BitTorrent is a so called *private tracker* [22]. A peer has to register with the tracker first, in order to download a file. The tracker counts the upload and download of every peer and bans peers who download more than they upload. We will analyse this *ratio* incentive in detail in Section III-B.

B. eMule

eMule [9] (and aMule² [2]); the open source alternative for all platforms, which is an extension of the eDonkey protocol, uses another approach for file-sharing than BitTorrent. According to the eMule protocol specification [19], a peer connects to an eMule server regardless of the files the client desires. The server provides the peer with file and peer indexing services. Therefore the peer sends a list of offered files to the server. eMule uses Kad [28], a Kademlia implementation, to allow file queries without being connected to a server or to find peers which are connected to another server. eMule generates two IDs: The client ID is used for communication with the eMule server and is therefore generated by the server, the user ID is unique per user and is used to identify a peer in different sessions. The user ID is chosen by the peer itself.

Like in BitTorrent, a file is divided into multiple pieces and this enables a peer to download from several peers simultaneously. If a peer wants to download a file it queries the server. If any of the peers known to the server has pieces of the desired file, the server sends a list of peers, which have the file, to the querier. If the peer is additionally or exclusively connected to the KAD network, the peer can query for a specific file hash to find more peers which offer the file or it can query the KAD network with keywords to obtain more file results. The peer than tries to download pieces from the peers it has known about.

The incentive mechanism implemented in eMule differs significantly from the BitTorrent tit-for-tat mechanism. Every peer maintains a waiting queue which contains peers waiting to download a piece of a file from this peer. All peers waiting in the queue of the uploading peer receive a score. This score is calculated from multiple values, such as the time it has waited in the queue, the priority of the file (defined by the uploading peer) and the history of the upload and download amounts of the downloading peer. A peer who has uploaded more to than he has downloaded will receive a higher credit. To identify a peer the unique user ID is used. This principal of couting the upload and download amounts, is similar to the already described ratio incentive, known from BitTorrent private trackers.

The peer with the highest score is on top of the queue.

²We will further on only talk about eMule. However, both names can be used synonymous.

The uploading peer will upload to the first peers in the queue, as long as each peer receives a minimum of 2.4 kB/s. Nonetheless, the number of peers a peer can serve is limited by the configuration of the eMule client. If another peer with a higher rating arrives, the uploading client will stop uploading to the the peer with the lowest score, until each peer receives the minimum upload rate. To give peers the change to share something, the score of new peers is set to a high value for the first 15 minutes.

III. INCENTIVE MECHANISMS

The incentive schemes of BitTorrent and eMule, presented in Section II, are just two ways of implementing incentive mechanisms in P2P-based systems. There exists a variety of different approaches on how to enforce cooperation and limit the ability of malicious peers to engage in attacks. We have classified incentive schemes into various classes, shown in figure 1.

The incentive mechanisms, we like to present, can be categorized according to how they give incentives to peers. There are *genuine incentives* and *artificial incentives* [29], the genuine incentives are directly implemented in the service and establish a link between current behaviour and future benefit. Genuine incentives are generally robust against rational behaviour. The artificial incentives, which tries to prove the cooperation between peers are implemented in an auditing process. Peers who seem not to cooperate may be punished afterwards. Artificial incentives are generally weaker than genuine incentives, since a peer may seem to cooperate, but in reality is behaving rationally.

Another important factor of incentive mechanisms is the *service maturation* [29]. A service may be granted instantly (*instantaneous maturation*), in the case of P2P file-sharing a peer may be able to download the file without fulfilling any demands. Otherwise, a service may be granted only after continuous interaction with the system (*progressive maturation*), like in BitTorrent or eMule, where a peer has to service other peers as well, in order to receive a service. Nielson et al. [29] show that only incentive mechanisms with progressive maturation are robust against rational attacks.

A. Genuine incentives

Genuine incentives can further be divided into soft schemes and hard schemes [25]. Soft schemes are based on a direct measurement of the cooperation e.g., the eMule score. Hard schemes are monetary based.

Soft schemes are *peer-approved* or *service-quality* oriented. In peer-approved incentive a peer may only serve peers with an equal or higher rating. This seems to require a global rating directory, but may be implemented by a tit-for-tat algorithm like in BitTorrent. Each peer is servicing other peers only if it has received service from that peer itself. A BitTorrent peer serves only peers who have uploaded to it, therefore BitTorrent can be classified as peer-approved.

In service-quality oriented schemes only the peers with the highest ratings are served. This matches exactly the definition of the eMule incentive mechanism, thus the eMule incentive mechanism can be classified as service-quality oriented. However, the BitTorrent incentive scheme also serves only the peers with the highest ratings (the best upload rate). According to this the BitTorrent incentive mechanism is also service-quality oriented.

Loginova et al. [25] states that service-quality schemes result in a better service. However, an optimal incentive mechanism can be implemented combining peer-approved and service-quality schemes. The statement that only a mixed incentive mechanism is optimal, is backed up by the observation of Nielson et al. [29], which states that BitTorrent is robust against rational attacks.

In hard schemes, the variety from monetary based incentives stretches from micro-payment schemes to a form of tokenexchange algorithms. Feldman et al. [10] and Park et al. [30] observed that monetary based incentive schemes allow the use of well investigated economical mechanisms from game theory. For P2P file-sharing applications, payment schemes are most interesting, since they can provide an incentive for uploading. In most payment schemes a payment infrastructure is needed. Until recently this has been a major disadvantage of monetary payment schemes. The need for a payment infrastructure has become unnecessary through decentralized payment algorithms. An algorithm which is interesting for P2P file-sharing applications is described by Feldman et al. [10]. A downloading peer pays for package delivery to the peer from which he receives the package. If intermediate peers are involved, each one increases the price of the package, so that everybody has a benefit from the delivery. Other payment algorithms are described in the same paper. Since they are not suitable for P2P file-sharing applications they are not described here.

B. Artificial incentives

In contrast to genuine incentives, artificial incentives do not directly participate in the service exchange [29]. They try to prove that there was a cooperation between peers. This is often done by an regularly executed auditing process or trust-based systems. The challenge of these auditing processes and trust-based systems is the lack of global knowledge in purely P2P-based systems [17].

Recently, there have been a lot of theoretical approaches on how a trust-based system can be established in a P2P environment [11], [12], [15], [17]. All these systems are based on the fact, that a peer will trust a neighbouring peer, if that neighbouring peer has provided a good service. A trust value is calculated based on the history between the peers and propagated throughout the network. This trust value is then used to regulate upload priority or other desired services within the P2P system [18], which effectively limits free-riding in P2P systems. Viswanath et al. [33] show that all these systems actually divide the network into social communities and this makes these systems vulnerable to Sybil attacks.

Auditing processes rely on a system in which a peer reports the service it has provided and received to either a central authority or other peers [29]. Through the comparison between the service provided to and service received from a peer, the auditing process can determine which peer is free-riding. The auditing of peers have to be simultaneously or at predefined points in time to avoid mismatching service reports.

Examples of such auditing processes are the private BitTorrent communities in which every peer has to report its upload and download amounts to the tracker [22]. In order to download from a private tracker, a peer has to register first. To avoid multiple accounts and prevent whitewashing, registration is only possible through invitations. The tracker creates a so called *ratio* for every peer. The ratio is the total upload amount divided through the total download amount. If the ratio is higher than one the peer has provided more service than he has received. If the ratio falls below a certain value the peer may be banned from that tracker.

The ratio forms an incentive to serve other peers. Liu et al. [22] have also shown that the ratio incentive effectively limits free-riding and that download speeds are significantly higher in private BitTorrent communities. This results from the fact that peers do not leave a swarm after they have finished a download, but stay as seeders to improve their ratio. Unfortunately, the ratio-based incentive is highly vulnerable to false upload and download reports and collusion attacks, which we will discuss in the following sections.

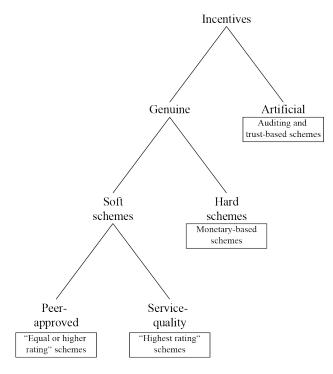


Fig. 1. Classification of incentive schemes

IV. ATTACKS AND COUNTERMEASURES

Open and heterogeneous systems like P2P file-sharing applications are always vulnerable to attacks performed by malicious peers. The motivations for attacks differ greatly. Some attacks may be performed due to ignorance of the

consequences, others may try to remove content from the network or to render the application unusable. In this section we will introduce classes of various attacks against P2P file-sharing systems and present countermeasures which effectively limit the ability to perform such attacks. We have summarized the attacks and countermeasures in table I.

A. Rational attacks

An attack class we mentioned in previous sections is the *rational attack* [29]. The rational attack is performed by free-riders and its motivation lies in saving peers' own resources, like upload bandwidth. A peer performing rational attacks acts *self-interested*. In case of copyrighted material, the motivation may lie in the avoidance of legal prosecution [3], [14]. A misconfiguration of the peers file-sharing application or router may also lead to a rational attack.

The incentives presented in Section III try to counter rational attacks. A well designed incentive can reduce free-riding to a minimum. The BitTorrent incentive mechanism was regarded as very robust against rational attacks [6]. However, recent studies have shown that the BitTorrent incentive mechanism does not prevent rational attacks [16], [23].

With the BitTorrent client *BitThief* [23] it is possible to download files from the BitTorrent network without uploading. This is even possible if the client only connects to leechers. *BitThief* does not use collusion or ID-based attacks, but rather exploits a weakness in the incentive mechanism design and uses the optimistic unchocking algorithm of BitTorrent to acquire pieces. To achieve high download rates *BitThief* connects to as many peers as possible. This increases the possibility of an optimistic unchocke and thus lets *BitThief* download without contributing to the system. Furthermore, *BitThief* does not report finished pieces to other peers, so peers will not request pieces from *BitThief*. Experiments have shown that through *BitThief* it is possible to download arbitrary files with no significant delay and without contributing at all.

A possible solution to this attack is to provide the abolishment of the optimistic unchoke and using of a different bootstrapping process. Levin et al. [21] have described a bootstrapping algorithm for BitTorrent, which solves the problem described above. A peer, which has not finished any pieces yet, requests a piece from another peer, the uploading peer encrypts the piece, sends it to the requesting peer with the instruction to upload it to another peer, which also has requested that piece from the initial uploader. After the third peer has received the piece, it reports to the initial uploader, that it has received the piece. The initial uploader then tells both peers the decryption key. If the requesting peer does not upload the peer to the third peer, it will not receive the key, thus having no advantage. The third peer also has no advantage if it falsely reports that it has not receive the piece, because he will not receive the decryption key either.

BitTorrent private trackers are suspect to *forged books* and *manufactured evident* attacks in their auditing process [29]. A peer reporting its upload and download amounts to the private tracker may just send fake values to increase its ratio, as the reports are normally not reviewed. Only very unusual

values can be manually detected. As a solution, the reporting algorithm can be modified. If a peer reports from whom he has downloaded and to whom he has uploaded instead of the total amount of upload and download, the tracker is able to review the data. This makes it much harder for peers to forge their ratios. Unfortunately, through collusion it is still possible to forge the ratio. We will discuss this and other attacks, which can be performed against BitTorrent, in the Subsections IV-C and IV-D.

eMule, on the other hand, needs further investigation on the robustness of its incentive mechanism. It is obvious that through the increased rating of a new peer there is a similar setting as with BitTorrent. This means, that it may be possible to bypass the eMule incentive mechanism through whitewashing.

B. Denial-of-Service attacks

Denial-of-Service (DoS) attacks try to overload a system so that no or insufficient service can be provided to the users. In most cases only malicious peers pursue DoS attacks, since there is no benefit for free-riders. Such attacks against P2P file-sharing systems become more difficult to implement, if a centralized server (like a BitTorrent tracker or eMule server) is available. However, if an attacker can disable these servers, peers will no longer be able to receive information on the location of files from these servers. In eMule a peer may just connect to another server or use the Kad network. However, in BitTorrent a tracker is usually responsible for a file. There may be no alternative than the DHT system to obtain information. If a private tracker is affected, DHTs are usually not available and the service will be interrupted for newly arriving peers. DoS attacks against the incentive schemes of BitTorrent and eMule are not known. An attacker can only disable single peers, but this brings him no advantage whatsoever.

DoS attacks usually overload a host by sending useless data to that host until it can no longer handle the load. In P2P networks, an attacker can make use of the queries within the network and trick the P2P network to perform attacks even against hosts outside the network [31]. In the case of Kademlia, Mysicka [28] has described attacks, which can be used to start DoS attacks. The attacker can insert information into Kademlia, by assigning itself a matching node ID, to redirect queries for files to an arbitrary host. This host then comes under attack by every querier.

Another attack against overlay networks, like Kademlia, is *churn* [29]. Multiple peers may frequently join and leave the network. Creating messages between the other peers until the network collapses.

Other DoS attacks include *accounting interruption* [29], which target artificial incentives. If a malicious peer can take out the auditing peer with a DoS attack. The auditing process can not be finished, effectively disabling the artificial incentive.

DoS attacks are usually difficult to *counter* [3]. If a host is under attack there is usually no way to stop an ongoing attack. The querying peers often act indirectly and unknowingly, thus punishment is not an option. DoS attacks by misusing DHTs have to be prevented by authenticating the information which are stored in the DHT. This can be achieved by establishing a

link between node ID, IP and Port, so that a node must have a matching ID for its IP and Port. This effectively limits the possible positions a node can occupy within the DHT.

Gupta et al. [11] has proposed *NeighbourTrust*, an artificial incentive, which can counter DoS attacks, in overlay networks like Kademlia, by assigning every peer a trust value. Peers with low values are not allowed to send multiple queries within the network, thus eliminating their ability to flood the network.

C. ID-based attacks

ID-based attacks include all attacks which are using ID manipulation or impersonation. P2P based system are always very vulnerable to ID-based attacks, since they lack a central authentication authority. A sophisticated ID generation decreases the ability of an attacker to perform ID-based attacks in general. BitTorrent, for example, limits ID-based attacks, since it uses the IP and port as the peer identifier. Therefore, the efficiency of the attack depends on the number of IP and ports available to the attacker.

The simplest ID-based attack is *whitewashing* [10]. This attack is only available in a system where IDs can be changed without or with low costs. An attacker can benefit from the system, change his ID and afterwards re-enter the system, thus allowing him to receive service without contributing to the system. In eMule this has only a limited effect, since a peer first has to gain a high score at other peers before he can benefit from the system. BitTorrent does not save the history with a peer and therefore whitewashing is not possible. However, private BitTorrent trackers use the ratio incentive and require a central ID [22]. If the ID creation is easy, whitewashing can render the ratio incentive useless.

Another possible ID-based attack is the *ID-mapping attack* [4]. In a system where a peer can choose its identifier freely (known as: zero-cost ID), an attacker can easily position himself in strategic positions within the network. Through ID mapping attacks it is possible to attack Kademlia [28].

The most prominent ID-based attack is the Sybil attack described by Douceur [8]. If the ID generation is not controlled by a central authority, an attacker can easily create multiple IDs for himself and use them to temper with the P2P system. eMule and BitTorrent are vulnerable to Sybil attacks [21], [28]. In eMule an attacker with multiple IDs can occupy multiple places in the upload queue of another peer, thus increasing the probability to be uploaded to. In BitTorrent the effects of Sybil attacks are more serious. A peer may create multiple IDs to increase the possibility for an optimistic unchoke. Furthermore, the attacker may upload with each of his IDs to a certain peer. If the upload rate for every ID is high enough, the peer is able to occupy all upload slots of that peer. This results in the peer only serving the attacker and a high download rate. The Sybil attack can be extended to an *eclipse attack* [3], [28]. There have been a lot of proposals on how to defend against Sybil attacks [15], [26], [33], [34]. Most of these countermeasures use artificial incentives, like distributed trust ratings. However, Dinger et al. [7] have stated that only a central ID creation can effectively prevent Sybil attacks. This leads to the assumption that private BitTorrent trackers are Sybil proof, if the ID creation is difficult enough. The Sybil based attack against BitTorrent can be countered, if the incentive mechanisms is changed, so that an uploading peer only uploads a proportional share to another peer, based on the peers own upload to the uploading peer. This gives peers which upload only a small amount of data a smaller download rate.

D. Collusion attacks

In a collusion attack a group of peers (or a peer controlling multiple IDs) work together to collectively gain an advantage. There exists an attack on BitTorrent described by Levin et al. [21], where multiple peers simultaneously stop uploading to a peer, from which they all are downloading from. They will receive a service without contributing until they are replaced with obedient peers by the optimistic unchocke algorithm. This attack can be combined with the Sybil attack. Thus a single peer can hijack the total upload of another peer with only contributing a minimum. Countering this attack is difficult. Best known countermeasure is the proportional share incentive proposal. With proportional share this attack will fail, if there exists a peer not colluding with the others. As mentioned in subsection IV-A, the ration incentive in private BitTorrent trackers is susceptible to collusion [22], even in a modified state. In most private trackers the minimum ratio is less than one. This means that a peer does not have to upload the same amount that he has downloaded. The minimum ratio is usually about 0.7, meaning that a peer has to upload only 70% the amount of his download. If a group of peers report traffic to each other that has never occurred, they gain free downloads from the private tracker. For example, a peer reports that he has uploaded and downloaded 70 GB from another peer. The other peer matches that report, bypassing the ability of the tracker to recognise the fraud. With a minimum ratio of 0.7 every peer has gained 30 GB of free download.

The trivial countermeasure to this attack is to increase the minimum ratio to one. However, this may be undesirable, since a high minimum ratio may be difficult to reach for every peer. Therefore, to counter this attack an *entropy scheme* has been proposed by Liu et al. [22]. The *upload entropy scheme* measures how broad the upload is distributed over the available peers. If a peer uploads only to a small group of peers, it is highly possible that this peer is colluding. The entropy scheme may not work if the number of colluders exceed a certain value. However, if the number of colluders is too big the system performance may decrease significantly. As with rational and DoS attacks, collusion attacks against eMule are not documented. Therefore, we can make no definite statements about collusion attacks against eMule.

V. DISCUSSION

Since the introduction of early P2P file-sharing systems implementing incentive schemes, many attacks against these schemes have been proposed. The case of BitTorrent has shown, that incentive schemes need permanent review and enhancements, otherwise they may fail to withstand newly

A CC 1	Attacks	Countermeasures
Affected system		
BitTorrent	Rational attack	replacement of optimistic un- choke algorithm
	DoS attack (tracker)	none
	Sybil attack	proportional share incentive scheme / artificial incentives
	Collusion attack	proportional share incentive scheme
BitTorrent private tracker	Forged books / manufactured ev- idence attack	changing the upload and download reports
	Whitewashing	restrictive ID creation
	DoS attack (tracker)	none
	Collusion attack	increasing minimum ratio / up- load entropy scheme
eMule	DoS attack (server)	none
	Whitewashing (limited)	none
	Sybil attack	artificial incentive schemes
Kademlia	DoS attack	artificial incentive schemes
	Churn	none
	ID-mapping	limiting ID space e.g., by
	attack	establishing a link between IP/Port and ID
	Eclipse attack	artificial incentive schemes
Artificial incentive schemes	Accounting inter- ruption	none

TABLE I SUMMARY OF ATTACKS

developed attacks. BitTorrent's greatest vulnerability is the optismistic unchoke algorithm which makes it possible to bypass the incentive scheme completely. However, the tit-for-tat incentive scheme is also not optimal, since it allows attackers to obtain multiple download slots. This is caused by BitTorrent's vulnerability to ID-based (Sybil) attacks and collusion attacks.

We have shown, that the BitTorrent protocol is not suitable for the requirements of private trackers and that a modified version of BitTorrent is needed. If the reporting of traffic is done in the current form, the used ratio incentive is useless. A single attacker can today easily forge his own ratio.

Since attacks against the eMule incentive scheme are rare, on today's knowledge it is safe to assume, that the eMule incentive scheme is robuster against rational attacks than the BitTorrent incentive scheme. A major vulnerablity known so far, is the generation of the user ID which open the eMule protocol to ID-based attacks, especially the Sybil attack.

The introduction of Kademlia in BitTorrent and eMule has extended their functionality and made them independent from centralized servers. However, Kademlia produces vulnerabilities, which allows malicious peers to perform more attacks than before. The insufficient authentication in Kamdelia is a major problem and needs to be fixed by the introduction of appropriate algorithms like artificial incentive schemes.

VI. CONCLUSION

In this paper we have analysed the incentive mechanisms of eMule and BitTorrent. We introduced an abstract classification of incentive mechanisms. Further, we have shown that there exists a great number of attacks against P2P file-sharing systems in general and their incentives. Most of these attacks can be countered through improvement or addition of already known incentives. This is especially true for BitTorrent, which has, until recently, been regarded as a robust P2P file-sharing system. However, there have been proposed numerous attacks against the BitTorrent incentive, which establish the need for improvement of the BitTorrent incentive mechanism. In the current form, the BitTorrent incentive mechanism can be regarded as broken. The same applies to the ratio incentive of BitTorrent private trackers, which, in its current form, is very easy to attack. The attacks against eMule are mainly based on the Kad network and attacks against the eMule incentive are less known and need further investigations.

REFERENCES

- [1] Eytan Adar and B.A. Huberman. Free riding on gnutella. *First Monday*, 5(10-2):2–13, 2000. I
- [2] The aMule Project. Retreived January 8, 2012, from http://www.amule.org. 2012. II-B
- [3] Prête Baptiste. Attacks on peer-to-peer networks. Dept. of Computer Science, Swiss Federal Institute of, pages 85–90, 2005. IV-A, IV-B, IV-C
- [4] Davide Cerri, Alessandro Ghioni, Stefano Paraboschi, and Simone Tiraboschi. Id mapping attacks in p2p networks. In Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE, volume 3, pages 6-pp. IEEE, 2005. IV-C
- [5] Kenjiro Cho, K Fukuda, and H Esaki. The impact and implications of the growth in residential user-to-user traffic. ACM SIGCOMM Computer, pages 207–218, 2006.
- [6] Bram Cohen. Incentives build robustness in BitTorrent. In Workshop on Economics of Peer-to-Peer systems, volume 6, pages 68–72. Citeseer, 2003. II-A, IV-A
- [7] J Dinger and H Hartenstein. Defending the Sybil attack in P2P networks: taxonomy, challenges, and a proposal for self-registration. In First International Conference on Availability Reliability and Security ARES06, volume 23, page 8 pp. Ieee, 2006. IV-C
- [8] J. Douceur. The sybil attack. Peer-to-peer Systems, pages 251–260, 2002. IV-C
- [9] The eMule Project. Retreived January 8, 2012, from http://www.emuleproject.net. 2012. II-B
- [10] Michal Feldman and John Chuang. Overcoming free-riding behavior in peer-to-peer systems. ACM SIGecom Exchanges, 5(4):41–50, 2005. III-A, IV-C
- [11] A Gupta, D Malhotra, and L K Awasthi. NeighborTrust: A trust-based scheme for countering Distributed Denial-of-Service attacks in P2P networks. 2008 16th IEEE International Conference on Networks, pages 1–6, 2008. III-B, IV-B
- [12] Jianli Hu, Quanyuan Wu, and Bin Zhou. Distributed and Effective Reputation Mechanism in P2P Systems. 2008 International Conference on Computer Science and Software Engineering, pages 110–115, 2008.
 III-B
- [13] Daniel Hughes. Free Riding on Gnutella Revisited: The Bell Tolls? The 2000 study A significant amount of free-riding occurs Free-riding is uniform across domains. *Ieee Distributed Systems Online*, 6(6):1–18, 2005. I
- [14] A.M.A. Huq. Can incentives overcome malicious behavior in p2p network? *Electronic Commerce*, pages 75–87, 2009. IV-A
- [15] Wen Ji, Shoubao Yang, and Bo Chen. A Group-Based Trust Metric for P2P Networks: Protection against Sybil Attack and Collusion. 2008 International Conference on Computer Science and Software Engineering, 3:90–93, 2008. III-B, IV-C
- [16] Seung Jun and M. Ahamad. Incentives in BitTorrent induce free riding. In Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems, pages 116–121. ACM, 2005. IV-A

- [17] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings* of the 12th international conference on World Wide Web, pages 640–651. ACM, 2003. III-B
- [18] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina. Incentives for combatting freeriding on P2P networks. *Euro-Par 2003 Parallel Processing*, pages 1273–1279, 2004. III-B
- [19] Yoram Kulbak and Danny Bickson. The eMule protocol specification. eMule project, http://sourceforge. net, 2005. II-B
- [20] James F. Kurose and Keith W. Ross. Computer networks: A top down approach featuring the internet, Solutions to Review Questions and Problems. Pearson Addison Wesley, 2004.
- [21] Dave Levin, K. LaCurts, Neil Spring, and Bobby Bhattacharjee. Bittorrent is an auction: analyzing and improving bittorrent's incentives. ACM SIGCOMM Computer Communication Review, 38(4):243–254, 2008. IV-A, IV-C, IV-D
- [22] Zhengye Liu, Prithula Dhungel, Di Wu, Chao Zhang, and K W Ross. Understanding and Improving Incentives in Private P2P Communities. In Proc of the 30th International Conference on Distributed Computing Systems IEEE ICDCS 10, (i), 2010. II-A, III-B, IV-C, IV-D
- [23] Thomas Locher, Patrick Moor, and Stefan Schmid. Free riding in BitTorrent is cheap. Proc. Workshop on Hot, pages 85–90, 2006. IV-A
- [24] Andrew Loewenstern. BitTorrent.org BEP 5: DHT Protocol. Retrieved December 20, 2011, from http://bittorrent.org/beps/bep_0005.html, 2008. II
- [25] Oksana Loginova, Haibin Lu, and X.H. Wang. Incentive schemes in peer-to-peer networks. *The BE Journal of Theoretical Economics*, 9(1):2, January 2009. III-A
- [26] Eng Keong Lua. Securing peer-to-peer overlay networks from Sybil Attack. 2007 International Symposium on Communications and Information Technologies, pages 1213–1218, 2007. IV-C
- [27] P Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. Peer-to-Peer Systems, 2002.
- [28] David Mysicka. eMule Attacks and Measurements. Dept. of Computer Science, Swiss Federal Institute of, 2007. II-B, IV-B, IV-C
- [29] S. Nielson, S. Crosby, and D. Wallach. A taxonomy of rational attacks. Peer-to-Peer Systems IV, pages 36–46, 2005. III, III-A, III-B, IV-A, IV-B
- [30] Jaeok Park and Mihaela van der Schaar. Pricing and Incentives in Peer-to-Peer Networks. 2010 Proceedings IEEE INFOCOM, pages 1–9, March 2010. III-A
- [31] Man Qi. P2P network-targeted DDoS attacks. 2009 Second International Conference on the Applications of Digital Information and Web Technologies, pages 843–845, 2009. IV-B
- [32] Hendrik Schulze and Klaus Mochalski. ipoque GmbH Internet Study 2008/2009. 2009.
- [33] Bimal Viswanath, Ansley Post, K.P. Gummadi, and Alan Mislove. An analysis of social network-based sybil defenses. In ACM SIGCOMM Computer Communication Review, volume 40, pages 363–374. ACM, 2010. III-B, IV-C
- [34] Haifeng Yu, M. Kaminsky, P.B. Gibbons, and Abraham Flaxman. Sybilguard: defending against sybil attacks via social networks. In ACM SIGCOMM Computer Communication Review, volume 36, pages 267– 278. ACM, 2006. IV-C

Virtual Coordinate Spaces

Beatrice Friess

Zusammenfassung—In dieser Seminararbeit wird das Lokationsproblem, sowie das Routing in drahtlosen Netzwerken betrachtet und es werden mögliche Lösungsansätze gegeben. Ein drahtloses Sensornetzwerk bezeichnet ein Netzwerk an Sen-

Ein drahtloses Sensornetzwerk bezeichnet ein Netzwerk an Sensorknoten, das Informationen sammelt und an eine Basisstation versendet. Sensorknoten werden Batterie betrieben und bestehen (im Normalfall) aus einem Sensor, einem Prozessor und einer drahtlosen Kommunikationseinheit.

Diese geringen Kapazitäten erschweren es ein Routing effizient durchzuführen. Um ein Routing mit globalen Positionen durchzuführen, in diesem Fall ein physikalisches Routing, wird mehr Hardware (z.B. ein GPS Sender) benötigt. Dadurch wird der Sensorknoten zu teuer und ein Konzept virtueller Räume wird verfolgt. Ein virtueller Raum wählt seine Koordinaten nicht anhand von physikalischen Daten und kann somit ohne Hardwareerweiterung durchgeführt werden.

Das Ziel dieser Seminararbeit ist es eine Einführung in virtuelle Räume zu geben, sowie verschiedene Verfahren zu vergleichen, die virtuelle Koordinaten zur Unterstzützung in drahtlosen Sensornetzwerken nutzen.

Weiterhin werden in dieser Seminararbeit die aktuellen Forschungsergebnisse von *Dhanapala et al.* [9] intensiver betrachtet, mit denen ein Routing auf virtuellen Koordinaten eingeführt wird, dass zudem eine Richtung angibt.

I. Einführung

Drahtlose Sensornetzwerke bestehen aus hunderten oder gar tausenden Sensorknoten. Diese Sensorknoten besitzen in ihrer Grundform einen Prozessor, einen Sensor und ein drahtloses Kommunikationsmodul. Die Sensoren sammeln Informationen ihrer Umgebung und verschicken diese über ein drahtloses Netzwerk an eine Basisstation.

Sensornetze werden in vielen Anwendungsgebieten, sowohl wissenschaftlich, als auch kommerziell, eingesetzt. Ein Beispiel hierfür ist das *Great Duck Island* [20]. Hier werden Sensoren für verschiedene Bereiche genutzt, zum Beispiel, um die Brutzeiten von Tieren zu beobachten. Sie werden aber auch zunehmend in alltäglichen Bereichen eingesetzt, unter anderem in der Industrie [17].

Die gesammelten Informationen werden über ein drahtloses multihop Netzwerk weitergeleitet. Hierfür muss ein Routing durchgeführt werden [3], [1].

Da Sensorknoten nur geringe Kapazitäten besitzen, muss der Energie- und der Speicherverbrauch dabei gering gehalten werden. Dies gilt bereits vor dem Routing, wenn der Sensorknoten seinen Standpunkt im Netzwerk bestimmt. Das Routing und die Bestimmung des Standortes sind somit die zentralen Probleme, mit denen sich verschiedene Papers befassen, die in dieser Seminararbeit betrachtet werden.

Um dieses Problem zu lösen gehen wir zuerst von einem physikalischer Ansatz aus. Dies bedeutet, man stützt sich auf Werte in einem physikalischen Referenzsystem, wie z.B. Positionen durch Satelliten. Durch diese physikalischen Koordinaten kann eine globale Position festgelegt werden.

Um den Standort des Sensorknotens zu bestimmen, greift man naturgemäß auf solche physikalische Daten zurück. Man könnte in der Praxis jeden Knoten mit einem GPS Sender ausstatten. Dies bringt aber eine Reihe von Nachteilen mit sich

Ein GPS Sender ist mit geostationären Satelliten verbunden, wodurch der Einsatz von GPS in Gebäuden, im Meer oder auch auf anderen Planeten problematisch ist. Zudem sind GPS Sender nicht zu gebrauchen, da die Sensorknoten energieeffizent, klein und kostengünstig gehalten werden müssen.

Diesem Problem widmen sich die virtuelle Koordinatenräume. Virtuelle Koordinaten in solch einem Sensornetzwerk können auf verschiedene Wege bestimmt werden ohne den Energieverbrauch zu steigern oder zusätzliche Hardware zu installieren. Dabei kann zum Beispiel der Winkel zwischen den verschiedenen Knoten genutzt werden, die Signalstärke oder auch die Hop Distanz.

Die Ansätze mit denen sich diese Arbeit intensiver beschäftigt, basieren dabei auf Hop Distanzen, um virtuelle Koordinaten zu berechnen. Als *Hop Distanzen* bezeichnet man die Anzahl der Weiterleitungen einer Nachricht über verschiedene Knoten bis zu einem Zielknoten.

Das Ziel dieser Seminararbeit ist es eine Übersicht über verschiedene Verfahren zu geben, die virtuelle Koordinaten anhand von Hop Distanzen bestimmen, sowie ein kurzer Einblick über andere Verfahren, die sich ebenfalls mit der Ermittlung von virtuellen Koordinaten befassen.

Zuerst folgt eine Einführung wichtiger Grundlagen in Abschnitt II, danach wird in Abschnitt II-A darauf eingegangen, welche allgemeinen Verfahren zur Bestimmung von virtuellen Koordinaten derzeit existieren. Anschließend wird in diesem Abschnitt auf die grundlegenden Eigenschaften eingegangen, die das ankerbasierende Verfahren mit sich bringt. In Abschnitt III wird dies vertieft, um darzulegen, welche Protokolle existieren, die mit Hop Distanzen ankerbasierend, virtuelle Knoten finden und über sie ein Routing ausführen. Dieser Abschnitt soll zudem noch eine Einführung in das gerichtete Routen auf virtuellen Koordinaten geben, basierend auf [9].

Abschnitt IV ist ein Diskussionsteil, der weiterführende Aspekte aus dem Bereich der virtuellen Koordinaten konkretisiert und einen Vergleich der gezeigten Verfahren liefert.

II. GRUNDLAGEN

A. Virtuelle Koordinaten

Die Hauptfrage für Sensornetze ist immer noch, wie sie am besten Informationen für ihre eigene Lage im Netz bestimmen und wie man ihnen Koordinaten zuteilt.

Formuliert wird dies als das klassische Lokationsproblem. Dieses ist definiert als Problem, bei dem es einen Knoten an einer Position in einem zwei- oder dreidimensionalen Raum gibt.

Diese Position soll nun möglichst genau bestimmt werden. Im Fall von Sensorknoten muss dies, aufgrund gering gehaltener Ressourcen, effizient durchgeführt werden. Aus diesem Grund wird hier in vielen Bereichen auf physikalische Koordinaten so gut es geht verzichtet.

Die Rahmenbedingungen, zum effizienten Bestimmen von virtuellen Koordinaten, können wie folgt betitelt werden:

- Wenig externe Infrastruktur
- Geringe Kommunikation
- Wenig Ressourcen
- Keinerlei externe Konfiguration

Nun können dafür zwei Ansätze gewählt werden. Der erste Ansatz beruht darauf nur spezielle Knoten, sogenannte Anker, mit physikalischen Koordinaten auszurüsten oder als Ausgangspunkt zu wählen, während der zweite Ansatz komplett auf diese sogenannten Anker verzichtet.

Ankerbasierende Verfahren

Die sogenannten Ankerbasierenden Verfahren beruhen darauf, dass eine ausgewählte Menge an Sensorknoten als feste Referenz gewählt wird, wonach sich das Sensornetzwerk richtet. Diese speziellen Knoten werden Anker oder Landmarks genannt.

Verschiedene Verfahren wählen eine unterschiedliche Anzahl von Ankern. Verfahren in [11], [10] wählen hierbei eine hohe Anzahl an Ankern, damit ein Knoten immer verschiedene Anker zur Verfügung hat und somit die eigene Position bestimmen kann.

Verfahren in [6], [4], nutzen eine geringe Anzahl von Ankern, die für das gesamte Netzwerk ausreichend sind und mit einer vorherigen Initialisierung als Orientierung genutzt werden.

Mit Hilfe solcher Anker werden nun auf verschiedenen Wegen virtuelle Koordinaten gesetzt. Hierbei gibt es die Möglichkeit über Zeitdifferenzen, Winkel des Signals, Signalstärke oder über Hop Distanzen die virtuellen Koordinaten zu wählen.

Allgemein ist das Vorgehen bei einem ankerbasierenden Verfahren wie folgt:

- Die Bestimmung des Abstandes zu verschiedenen Ankerknoten (zum Beispiel über Hop Distanzen)
- 2) Setzen der eigenen Position anhand der gesammelten Daten
- Nutzen lokaler Heuristiken, um die eigene Position zu verbessern

Beim Entfallen der letzten Punktes spricht man von einem zweiphasigen Algorithmus. Ein Beispiel für solch einen Algorithmus liefert *ad-hoc positioning* [16]. Bei diesen Verfahren kann es zu vielen Ausreißern kommen, wie in [13] beschrieben.

Ein dreiphasiger Algorithmus integriert diesen letzten Punkt. Als Beispiele können hier *robust positioning* [18] und *n-hop multilateration* [19] genannt werden.

Diese beiden Verfahren basieren auf einer leicht veränderten ersten Phase zu Bestimmung des Abstandes.

Bei ankerbasierenden Verfahren sind zwei Hauptprobleme zu nennen, die von Protokollen verschieden gelöst werden. (Das genaue Vorgehen der verschiedenen Protokolle wird in Abschnitt III weiter erläutert) Konsistenz:

Um die Konsistenz zu wahren, müssen zwei Punkte beachtet werden.

In erster Linie sollte beachtet werden, dass Knoten, die nahe beieinander liegen auch ähnliche Koordinaten zugewiesen bekommen.

Zusätzlich müssen nicht benachbarte Knoten einen Mindestabstand einhalten, um eine Faltung zu vermeiden. Die Faltung stellt hierbei ein weiteres Problem dar.

Von Faltung spricht man, wenn zwei Knoten, die eigentlich weit voneinander entfernt sind, ähnliche Koordinaten zugewiesen bekommen und das Gebiet zwischen diesen beiden Knoten *gefaltet* wird. Beim anwählen von Sensorknoten in einem bestimmten Gebiet, bei z.B. einer Notfallmeldung bei einem Erdbeben, bei dem Menschen evakuiert werden müssen, kann es dann vorkommen, dass die Notfallmeldung in ein Gebiet übertragen wird, dass physikalisch gesehen weit entfernt liegt und somit Menschen evakuiert werden, die nicht betroffen sind.

Ankerfreie Verfahren

Ankerfreie Verfahren werden zum Beispiel in [13] eingeführt. Ein Beispiel für solch ein Verfahren ist das Anchor-Location-Free Verfahren. Dieses basiert darauf, dass zwei Ankerpaare gesucht werden, die jeweils mit einer Geraden verbunden werden. Das Ziel ist es, dass diese zwei Geraden sich orthogonal schneiden und ein Koordinatensystem bilden.

In [13] wurde anhand von verschiedenen Simulationen gezeigt, dass dieses Verfahren, im Gegensatz zu den ankerbasierenden Verfahren, das zufriedenstellendere Ergebnis ergibt. Um dieses zu verbessern wurde hierbei das Konzept des ankerfreien Verfahrens um das Schema von Clustern erweitert. Durch ein Routing über Clustersysteme zeigt sich hierbei das ankerfreie Verfahren als sehr effizient, jedoch ist dieses Verfahren bisher wenig erforscht und wird in dieser Seminararbeit nicht weiter vertieft.

Weiteres hierzu ist in [14] nachzulesen.

B. Topology Maps

Probleme, die in virtuellen Koordinatenräumen auftreten sind stark mit dem Problem der fehlenden Richtungen verknüpft, z.B. das Auftreten von lokalen Minima und weitere Probleme, die ihren Ursprung in den fehlenden geografischen Informationen haben.

Diese Probleme kann man lösen, in dem man physikalische Daten mit virtuellen kombiniert. Damit kann nicht nur das Problem der virtuellen Daten, sondern auch das Problem der Richtung in einem Netzwerk aufgelöst werden.

In dem Paper [8] wird eine theoretische Basis und Techniken dazu geliefert, wie es möglich ist eine 2-D physisches Topologie Map zu erhalten. Diese soll ein gerichtetes Kartesisches Koordinatensystem beinhalten und eine Vermeidung von geografischen Voids.

Topologie Maps basieren auf einem Verfahren ähnlich der Eigenwertbestimmung (Singulärwertzerlegung). Die neue Methode generiert eine zwei-dimensionale Topologie Map

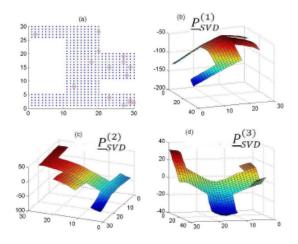


Abbildung 1. a)ursprüngliches Netzwerk b)- d)nach 3 Anwendungen der Topologie [8].

(TPM) mit Hilfe eines Sets an M Ankern.

Die TPM bewahren die Informationen aus der Nachbarschaft eines Knotens auf, aber sie speichern nicht die Koordinaten oder die Position aus der alten Topologie, genauso wenig, wie die Abstände aus der alten Topologie. Mit geografischen Informationen können diese Topologien auch zur Lokalisierung genutzt werden.

Die Abbildung 1 zeigt, dass die Topologie Map nicht linear skaliert ist aber rotiert mit der ursprünglichen Topologie übereinstimmt. Dadurch sind die ursprüngliche Topologie und die neue topologisch isomorph.

Durch bessere Ankersetzung kann das Ergebnis noch verbessert werden, was weiteren Abbildungen aus [8] zu entnehmen ist.

III. VIRTUELLE KOORDINATEN MIT HOP DISTANZEN

Der nachfolgende Teil befasst sich mit Protokollen, die virtuelle Koordinaten mit Hilfe von Hop Distanzen wählen. Diese Protokolle sind unabhängig voneinander zu betrachten.

A. Virtuellen Koordinaten für connectivity-based Protokolle

Einer der frühesten Ansätze zum Bestimmen von virtuellen Koordinaten, basierend auf Hop Distanzen, wird von *Moscribroda et al.* in [15] beschrieben.

Zuweisung

Der in [15] vorgestellte Algorithmus zeichnet sich dadurch aus, dass er für GPS Daten erweiterbar ist, um die Lokalisierung zu verbessern, aber auch ein Mapping beinhaltet, dass eine Integration der Abstände zwischen den Knoten möglich macht (ein Punkt, der allgemein sehr teuer ist).

Die Ideen des Algorithmus werden wie folgt zusammengefasst:

Im ersten Schritt wird der Abstand zwischen zwei Knoten bestimmt (mittels Hop Distanzen). Diese Abstände werden in einer Distanzmatrix gespeichert und später benutzt, um die Knoten in einen n-dimensionalen Raum zu positionieren. Durch das Setzen der Knoten wird das *freie Set* (das Set an Knoten, das bisher gesetzt wurde) ziemlich groß, da die Dimensionen immer weiter steigen. Um dieses wieder zu vereinfachen, wird dieses n-dimensionale Set auf ein 2-dimensionales reduziert (siehe Algorithmus 3 in [15]). Im weiteren Verlauf werden einige Punkte auf Rasterpunkten sitzen und die anderen Punkte um diese herum gesetzt.

In der Praxis müssen nun verschiedene wichtige Punkte betrachtet werden:

Routing

Durch das reduzieren auf zwei Dimensionen kann es vorkommen, dass einige Knoten dieselben Koordinaten besitzen, somit kann hier kein normales *geometric greedy routing* [12] mehr genutzt werden. Um die möglichen Ausfälle zu verringern gibt es zwei mögliche Ansätze.

Die erste Methode wäre die Knoten kurzzeitig zu stören, damit sie mit minimaler Verschiebung nebeneinander liegen. Unter anderem leidet hierbei aber die Qualität der virtuellen Koordinaten und Sackgassen können erneut entstehen.

Eine weitere Methode ist ein Neighbour-of-Neighbour Routing, dass an dieser Stelle effizienter ist. Beim Neighbour-of-Neighbour Routing wird in einem 2-Hop Umkreis der nächste Hop gewählt (dieses Verfahren kann trotzdem fehlschlagen).

B. Virtual Coordinate Assignement Protocol (VCap)

Die Annahme für das Vcap Protokoll [5] ist, dass es sich um ein großes Netzwerk handelt und die Knoten keinerlei Informationen besitzen, was ihre eigene Position betrifft und wie groß das Netzwerk ist. Die Knoten sind statisch oder quasi-statisch und besitzen eine eindeutige ID und einen festen Übertragungsradius. Es wird hier ebenfalls die Methodik der Hop Distanzen verwendet.

Zuweisung

Es werden vier Runden gebraucht, um drei Ankerknoten zu selektieren. Jeder Knoten bekommt dann ein Trippel von Daten mit den Hop Distanzen zu den Ankern.

Die vier Election Phasen sollen hier genauer betrachtet werden:

Das senden der W_SET Nachricht ist die erste Übertragung in dem Netzwerk. Hierbei wird jedem Knoten eine Nachricht mit dem Hop Count 1 geschickt. Bei jedem Knoten wird dieser Count um 1 erhöht. Wenn mehrere Nachrichten mit einem Hop Count eintreffen, wird der niedrigste Wert verwendet und die Nachricht mit dem höheren Hop Count verworfen und nicht weitergeleitet.

Die weiteren Election Nachrichten (X, Y, Z) werden analog zueinander ausgeführt. Nach einem Timeout wird die X_ELECT Nachricht (bzw. Y oder Z) gesendet. Diese dienen dazu die virtuellen Koordinaten zu setzen. Zuletzt besitzt jeder Knoten ein Trippel mit den Koordinaten X, Y, Z.

Die Anker X, Y, Z werden so gewählt, dass sie möglichst weit auseinander liegen. Aus diesem Grund werden die Koordinaten nacheinander durch eine Election Nachricht gewählt, um die bereits gesetzten Koordinaten zu berücksichtigen.

Die W Nachricht ist zu Beginn nur im Gebrauch, um die Knoten mit der größten Entfernung zu suchen, danach ist W nicht mehr interessant.

Routing

Hier kann nun ein geografisches Routing Protokoll ansetzen.

C. Virtual Domain and Coordinate Routing (VDCR)

In [11] wird ein weiteres Protokoll eingeführt. Wie in Vcap gilt auch hier, dass die Knoten statisch oder quasi-statisch sind und somit eine eindeutige ID besitzen und einen feste Übertragungsradius.

In VDCR werden einige Ankerpunkte gewählt, die untereinander einen kürzesten Pfad über Routing finden können. Für jeden Knoten wird eine Liste mit dem Vorgänger und dem Nachfolger gespeichert, zuletzt wird dies als Tree abgespeichert.

VDCR wird in zwei Protokolle aufgeteilt:

Das erste Protokoll ist das *Zuweisungsprotokoll*, dieses wird bei der Initialisierung des Netzwerkes ausgeführt, um die virtuelle Domain zuzuweisen (der Bereich, in dem der Tree sich befindet) und danach, um die virtuellen Koordinaten in diesem Raum zuzuweisen.

Das zweite Protokoll ist das *Routingprotokoll*. Hierbei gilt, je mehr Sensorknoten vorhanden sind, umso effektiver arbeitet das Routingprotokoll.

Zuweisung

Um das Routing ohne physikalische Informationen durchführen zu können, werden sowohl die Virtual Domain, als auch die virtuellen Koordinaten verwendet. Für das Finden der nächsten Knoten werden hierbei die virtuellen Koordinaten verwendet und für das Lokalisieren des kürzesten Pfades wird hierfür die Virtual Domain genutzt.

Allgemein kann ein Knoten selbst herausfinden, ob das Ziel in der eigenen oder einer anderen Domain liegt.

Die Zuweisung geschieht in 3 Phasen, in denen die Nachrichten bis zu den Blättern geroutet werden. Von den Blättern aus wird ein Counter zurück zur Wurzel geschickt, um den Hop Count zu den Blättern zu bestimmen. Damit werden die virtuellen Koordinaten gesetzt.

Die Information über die virtuelle Domain beinhaltet alle virtuellen Koordinaten, also alle Koordinaten der Unterknoten, die sich in dieser virtuellen Domain befinden. Diese Informationen werden an alle Knoten in dieser Domain geroutet. Dadurch wird die virtuelle Domain zugewiesen.

Routing Ausgehend davon, dass ein Knoten das Wissen über seine Domain und seine Koordinaten kennt, kann wie folgt ein neues Protokoll entwickelt werden. Um die versendeten Datenpakete klein zu halten, werden keine Informationen über die Domain oder die Koordinaten in die Datenpakete aufgenommen. Diese Daten werden nur für das Routing gebraucht und somit auch nur verwendet, wenn ein Routing vonnöten ist.

Beim Routen wird nun wie folgt vorgegangen: Ein Quellknoten sendet ein *path-discovery* Paket mit den Informationen der Domain und der virtuellen Koordinate

des Zieles weiter. Wenn ein Knoten dieses Paket empfängt, wird es an den nächsten Hop weitergereicht, der dem Ziel am nächsten ist. Der Knoten, der dieses Paket weitergereicht hat, speichert diese Informationen mit einer ID in seiner temporal forwarding table, um später das Routing schneller durchführen zu können.

Das Routing Protokoll beinhaltet drei Routing Modi:

Einen Domain Modus, einen Koordinaten Modus und einen Angle-based Landmark Modus.

Der Domain Modus führt das Routing in der Domain durch und wählt zum Finden des Zielknoten eine Domain aus. Wenn in dieser Domain der Zielknoten nicht gefunden werden kann, wird die nächste Domain gesucht. Wenn dies ebenfalls fehlschlägt, greift der Koordinaten Modus.

Im Koordinaten Modus wird die Distanz über die virtuellen Knoten berechnet und der Hop gewählt, der die kürzeste Distanz zum Ziel besitzt.

Da trotz dem Koordinaten Modus noch immer Sackgassen entstehen können, wird noch ein Modus aufgeführt, um dies zu verhindern. Der Angle-based Landmark Modus ist hierbei ein aufwendiger Algroithmus, dieser wird in [11] ausführlich behandelt.

D. Beacon Vector Routing (BVR)

Der BVR Algorithmus, eingeführt von Fonseca et al. in [10], definiert ein Set an Koordinaten und eine Distanzfunktion. Diese Koordinaten stehen in Referenz zu einem kleinen Set an Beacon Knoten. Dies ist ein zufälliges Set an Knoten, dass einen einfachen standard reverse path tree construction algorithm benutzt. Jeder Knoten kennt seine Distanz zu jedem der Beacon Knoten. Die Koordinaten dieses Knoten sind die Hop Abstände in einem Vektor. Wenn das greedy forwarding mit diesen Koordinaten fehl schlägt, wird ein Korrekturverfahren angewandt, um die Übertragungsgarantie sicherzustellen. Hierbei ist es wichtig, über Beacon Knoten hinwegzugehen, da es meistens besser ist sich auf diese Knoten zuzubewegen, als sich zu entfernen. Sich entfernen bedeutet oftmals sich in die falsche Richtung zu bewegen. Von einem Beacon Knoten erreicht man das Ziel ohne Probleme.

Vorgehen beim Routen:

- Aktualisieren der Header Informationen
- Greedy Forwarding ausführen
- Wenn dieses fehlschlägt, Fallback Mode ausführen
- Wenn dieses auch fehlschlägt, Scoped Flood benutzen

Fallback Mode

Wenn es keinen Nachbarn mehr gibt, der einen niedrigeren Wert hat, wird ein Fallback Mode zur Korrektur verwendet. In diesem Fall wird das Paket Richtung Beacon Knoten zurückgesendet zu einem Parent Knoten und von dort aus wird erneut das Greedy Forwarding angewendet.

Scoped Flood

Wenn das Paket an seinem Beacon Knoten angekommen ist, ist das Greedy Forwarding fehlgeschlagen und es kann kein Fallback Mode mehr eingesetzt werden. In diesem Fall wird ein scoped Flood angewandt. Vom Beacon Knoten aus wird das Paket überall hin gesendet, scoped ist die Begrenzung

(der Hop Count), wie weit die Nachricht gesendet werden soll.

Jeder Eintrag im Beacon Vector hat eine Sequenznummer. Diese Sequenznummer muss periodisch aktualisiert werden. Wenn keine Aktualisierung passiert, wird der Beacon Knoten aus dem Vector gelöscht.

Wenn zu viele Beacon Knoten gelöscht werden, bestimmen einige Knoten sich selbst als neuen Beacon Knoten, dies geschieht, wenn eine konfigurierbare Grenze überschritten wird. Bevor der Knoten ein Beacon Knoten wird, wird ein Timer aktiviert, für den Fall, dass ein anderer Knoten bereits als Beacon Knoten eingesprungen ist.

Genauso funktioniert dies in die andere Richtung. Wenn festgestellt wird, dass die maximale Anzahl an Beacon Knoten erreicht wurde, wird der Knoten mit dem höchsten Identifizierungswert wieder als normaler Knoten gesetzt. Durch das ablaufen der Sequenznummern übernehmen andere Knoten automatisch diese neue Einstellung.

E. Logical Coordinate Routing (LCR)

Das Logical Coordinate Routing wird in [4] vorgestellt. Allgemein basieren diese logischen Koordinaten auf dem klassischen Vektorprinzip mit Hop Distanzen.

Zuweisung

Das Vorgehen kann dabei wie folgt beschrieben werden: Die gewählten Landmarks senden ein Beacon Signal an alle Knoten in dem Sensornetzwerk aus, dieses *Beacon* Signal enthält einen Hop Count Parameter. Die Knoten speichern die kürzeste Hop Distanz zu dem Landmark. Für jedes Landmark speichern die Hops einen Wert in einem Vektor. Bei vier Landmarks ergibt sich somit ein 4-dimensionaler Vektor. Die Landmarks speichern in ihrem Vektor, den eigene Abstand mit 0 ab und sind so als Landmarks identifiziert..

Dadurch entstehen auch Inkonsistenzen:

Erstens: Andere Koordinaten könnten Nullwerte enthalten, da Nachrichten verloren gegangen sind, aber sie repräsentieren keinen Landmark, also beinhalten sie fehlerhafte Koordinaten. Diesem Problem wird entgegen gewirkt, indem der Knoten anhand seines Nachbarn den Wert setzt. (Nachbar +1).

Zweitens: Der Nachbar hat mehr als einen Hop Count Unterschied.

In diesem Fall gleicht sich der Knoten mit dem höheren Count an den Knoten mit dem niedrigeren Count an. Danach wird ein neues Signal gesendet, damit sich die Nachbarn gegebenenfalls anpassen können.

Routing

Die Knoten wissen nun über ihre Nachbarn Bescheid und können ihre eigenen Hop Distanzen mit denen der Nachbarn vergleichen. Auf diese Art und Weise ergibt sich ein lokales Vergleichssystem, mit dem es den Knoten möglich ist zu bestimmen, in welche Richtung ein Paket (im greedy forwarding) gesendet werden soll.

Durch dieses Verfahren können auch Schleifen entstehen, die verhindert werden müssen. Dazu implementiert der Algorithmus eine *Loop Avoidance*.

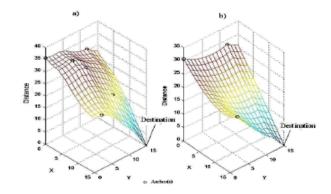


Abbildung 2. Visualisierung von CSR [6].

Im Falle eines Dead-End werden Pakete an den letzten Knoten (mit höherem Hop Count) zurückgesendet. Von dort aus wird dann versucht einen anderen Weg zu finden, um das Paket an das Ziel zu routen. Dies wird als *Void Avoidance* bezeichnet.

F. Convex Subspace Routing (CSR)

CSR [6] agiert auf einem logischen Distanzraum. Die Besonderheit an CSR ist, dass es sich ein Subset von Ankerknoten wählt und dann über einer konvexen Oberfläche routet. Dies erhöht die Routability, besonders in lokalen Bereichen. Eine Distanzfunktion ist dann konvex, wenn es ihr Set ebenfalls ist.

Zuweisung

Auf Abbildung 2 ist zu sehen, wie drei Ankerknoten eine konvexe Fläche bilden, während dies bei mehreren Ankerknoten nicht möglich ist. Für das ideale Routen muss nun die Quelle und die Senke auf der konvexen Oberfläche liegen. Dadurch werden lokale Minima weitestgehend verhindert.

Eines der Probleme hierbei ist es diese Ankerknoten so zu wählen, dass gut gelegene Knoten zum Routen auf diesem konvexen Netz liegen. Weitere Informationen zu Problemlösungen, siehe [6].

Routing

Der Algorithmus arbeitet mit Ankertripletts. Wenn ein Paket ein lokales Minimum erreicht, ist der Zielknoten nicht im eigenen Triplett, also muss er im Nächsten sein, das Paket wird an das nächste (naheliegende) Triplett weitergegeben. Bei doppelten Koordinaten wird am Zielpunkt (Hop Count 0) die ID des Knoten mit dem des Paketes verglichen (Ziel ID). Wenn die ID nicht übereinstimmt, wird im nächsten Triplett erneut nach dem Zielknoten gesucht. Die verschiedenen Tripletts sind indexiert, somit werden sie gekennzeichnet als 'nicht erfolgreich', wenn in diesem Triplett bereits das Routen des aktuellen Pakets fehlgeschlagen ist.

Um keinen Fehlschlag beim Routing zu haben ist eine der Anforderungen für das CSR, das jeder Knoten in einem Triplett liegt. Der Algorithmus kann in diesem Fall aber auch erweitert werden, um externe Ankerknoten zu integrieren. Ergebnisse der Simulation zeigen, dass CSR eine signifikante Verbesserung zeigt in der Routability und der Energieeffizienz. Besonders im Vergleich zu LCR zeigt CSR wie man mit weniger Hops eine bessere Routability erreichen kann. (durch weniger Hops fällt auch der Energieverbrauch stark ab)

Konvexe Verfahren, die mehr als drei Ankerknoten benutzen, sind zur Zeit noch nicht untersucht, drei Ankerknoten bilden hierbei das Minimun, um einen konvexen Raum aufzuspannen.

G. Directional Virtual Coordinate Routing (DVCR)

In diesem Bereich werden nun die wichtigsten Aspekte für das gerichtete Routen in virtuellen Koordinatenräumen eingeführt und erläutert.

Dimensionsreduzierung und Ankerevaluierung

Das Ziel des Papers [7] ist es die Dimensionen von virtuellen Koordinaten zu reduzieren und damit eine höhere Energieeffizienz zu erreichen, ohne dabei die *routability* zu verringern. Das Problem der Ankersetzung wird hierbei als wichtig deklariert. Das Problem der Evaluation von Ankern ist, dass einige Anker falsche Informationen beinhalten könnten oder redundante Informationen, genauso wie nicht komplette Informationen.

Das Problem kann mit zusätzlichen Ankern beseitigt werden, aber dies bedeutet auch, dass höhere Energiekosten zustande kommen. Ein neuer Anker kann nun genutzt werden, um:

- Ein gutes Subset zu finden, dass die verbesserte *routability* erhält
- Bestimmen eines Terminierungskriteriums, um neue Anker in ein Netzwerk zu integrieren
- Gute Positionen für Anker zu bestimmen

Dimensionreduzierung in virtuellen Koordinatenräumen

Das bestehende Problem ist das Finden guter Positionen für Anker in einem großen Netzwerk. Die verschiedene Anzahl an gewählten Ankern ist gleichzeitig der ausschlaggebende Punkt für die Effektivität des Routings.

Um einen optimalen Anker zu finden, können verschiedene ausprobiert werden. Dies ist jedoch sehr kostspielig und in WSN nur begrenzt möglich. Wenn dies zentral geschieht, könnte mit einem großen Set an Ankern gestartet werden und die Dimension daraufhin reduziert werden.

Das SVD Verfahren vollzieht nun hier die Komprimierung der Dimension. Das Problem, dass sich hier nun stellt ist, wie weit sollte die Dimension reduziert werden? Und wie weit kann sie das, ohne dass die *Routability* zerstört wird?

Die Lösung dieses Problems ist ein Schwellenwert, der nicht unterschritten werden darf.

Gerichtete virtuelle Koordinaten Räume

Um gerichtete virtuelle Koordinaten [9] zu erhalten, kann nun wie folgt vorgegangen werden.

Transformation im 1-D Raum

Es wird angenommen, dass die Knoten wie in Abbildung 3 verteilt sind. Durch eine neu definierte Funktion (siehe [9]) werden Werte geschaffen, die eine Richtung bestimmen. Das

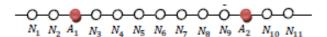


Abbildung 3. Transformation im 1-D Raum [9].

EXAMPLE	VC T	RANSI	FORM	ATIO	N STE	PS IN	1DN	TETW	ORK.	SHO	WN D	N FIG.	1
NODE ID	N_1	N ₂	A_1	N_3	N_4	N ₅	N_6	N ₇	<i>N</i> ₈	N ₉	A_2	N ₁₀	N ₁₁
h_{iA_1}	2	1	0	1	2	3	4	5	6	7	8	6	10
h_{iA_2}	10	9	8	7	6	5	4	3	2	1	0	1	2
$h_{iA_1} - h_{iA_2}$	-8	-8	-8	-6	4	-2	0	2	4	6	8	8	8
$h_{iA_1} + h_{iA_2}$	12	10	8	8	8	8	8	8	8	8	8	10	12
$f(h_{iA_1}, h_{iA_2})$	6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6

Abbildung 4. Tabelle der Gewichtungen [9].

Ergebnis ist aus der Tabelle in Abbildung 4 abzulesen.

Ergebnis: Das Problem des lokalen Minimums wird komplett umgangen.

Transformation im 2-D Raum

Hierbei wird als Grundlage ein Vektor genommen mit den Hop Distanzen und nicht mehr eine einzelne Hop Distanz. Die bekannten Vektoranwendungen können hier nun genutzt werden.

Folgende Eigenschaften treten nun auf:

- Knoten haben einen positiven oder negativen Wert, um die Richtung zu bestimmen. Ein Knoten, der den selben Abstand zu Anker 1, wie zu Anker 2 hat, besitzt den Wert 0 (er liegt direkt in der Mitte).
- Durch die Wahl von mehr Ankerknoten kann das System in ein kartesisches Koordinatensystem aufgesplittet werden.

Routing in einer gerichteten virtuellen Domain

An dem folgenden Beispiel wird nun gezeigt, dass das Routen im gerichteten virtuellen Raum eine 100 prozentige *Routability* erreichen kann.

Für dieses Beispiel wird ein Constrained Tree (CT) genommen, der den Grad 3 besitzt.

Ein CT mit dem Grad 3 ist ein Baum, der nur einen Zweig jeweils besitzt. Die Zweige haben keine Unterzweige und es gibt auch keine doppelte Verzweigung an einem Punkt.

In einem CT gibt es folgenden Eigenschaften:

- Der Abstand zwischen zwei Adjazenzknoten in einem Zweig ist konstant und basiert auf einem Kreuzungsknoten. Der Abstand kann somit genauestens angegebenen werden.
- Nur für den Hauptstrang (Backbone) gilt, dass der Single Abstand zwischen einem Knoten N und A1 + der Abstand zu N und A2 = der Abstand zwischen A1 und A2 ist.
- In einem CT kann der Verzweigungsknoten alle Mitglieder des Zweiges identifizieren und weiß, wie viele Hops Abstand diese von dem Verzweigungsknoten entfernt liegen.

100% Routbarkeit kann gegeben werden, wenn folgendes Schema verfolgt wird.

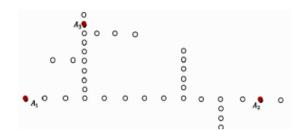


Abbildung 5. Erweiterter CT [9]

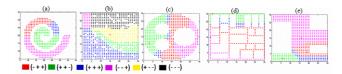


Abbildung 6. Ergebnisse aus [9]

Ein Paket wird über den Backbone geroutet und jeder Verzweigungsknoten entscheidet, ob das Paket zu einem Zielknoten auf seinem Ast geroutet werden soll.

Wenn dies nicht der Fall ist, wird das Paket auf dem Backbone weitergereicht. Da es sich hierbei um gerichtete Koordinaten handelt und das Paket somit die Richtung kennt, in die es geroutet werden muss, erreicht es sein Ziel zu 100%.

Dabei sollten die zwei gewählten Ankerknoten für den Backbone so gewählt sein, dass alle Verzweigungen zwischen ihnen liegen (wegen der Bestimmung der Richtung). Dabei müssen die beiden Ankerknoten aber keine Enden darstellen.

Bei einem Zweig, der einen weiteren Zweig beinhaltet (siehe Abbildung 5) ist Routing ebenfalls möglich. In diesem Fall wird ein weiterer Ankerknoten benötigt, der auf dem ersten Zweig existiert und somit als sub-tree das Routing genauso ausführt, wie es bereits auf dem Backbone der Fall war. Auf diese Art und Weise kann jedes Routing durchgeführt werden, ohne dass Zyklen entstehen.

Für den Fall, dass ein Verzweigungsknoten zwei Äste hat, kann der Verzweigungsknoten den Koordinaten einen weiteren Wert hinzufügen, um zu unterscheiden, ob es der obere oder der untere Knoten ist, den er ansprechen muss. Analog kann dies auch über weitere Grade hinaus formuliert werden.

Simulationsresultate siehe Abbildung 6

Ankerwahl und Topologie Maps

In diesem Abschnitt wird eine Erweiterung vorgestellt für das gerichtete Routing im virtuellen Koordinaten Raum, wie es in [2] beschrieben wird. Das Verfahren hierzu trägt den Namen *Extrem Node Search (ENS)*. Bei diesem Verfahren wird das Ergebnis verbessert, in dem Extremalknoten als Ankerknoten gewählt werden.

Ein Extremalknoten ist ein Knoten, der ein lokales Minimum oder Maximum bildet, in einem gerichteten virtuellen Koordinatenraum mit zwei Ankerknoten.

Das Vorgehen von ENS kann wie folgt beschrieben werden:

 2 Anker werden zufällig ausgewählt, die das Netzwerk fluten, um ein VCS erstellen

- Jeder Knoten generiert lokal seinen gerichteten virtuellen Koordinatenraum
- Jeder Knoten testet, ob ein lokales Minimum oder Maximum vorhanden ist
- Die ausgewählten Ankerknoten mit den lokalen Minimum oder Maximum bilden das neue VCS

Die zuvor ausgewählten zufälligen Anker können zusätzlich zu den Neuen gewählten werden, um die Routbarkeit zu erhöhen, wenn nötig.

Das Ergebnis ist, dass nun ein Grad zwischen den Knoten entsteht, der zum Testen genutzt werden kann. Dieser wird zusätzlich genutzt, um eine TPM zu erstellen.

Somit ergeben sich die folgenden Versionen, die verglichen werden können:

- TPM mit der Benutzung von orthogonalen Daten aus DVC (Grad zwischen den Knoten genutzt)
- SVD basierte TPM, die mit Extreme Note Search Ankern arbeitet
- SVD basierte TPM mit 10 zufällig generierten Knoten

IV. DISKUSSION

In diesem Abschnitt wird ein erweiterter Bereiche der virtuellen Koordinaten betrachtet. Der Abschnitt dient dazu den Blick auf das Themengebiet zu erweitern.

Zusätzlich wird ein Vergleich der verschiedenen Verfahren noch durchgeführt.

A. Szenarien Auswahl von Simulationen

Ein großes Problem im Bereich Sensornetzwerke stellen die oftmals zu naiv gewählten Ansätze da, anhand denen die neuen Algorithmen getestet werden. Schon bei kleineren Abschweifungen, scheitern die meisten Protokolle.

Dies wird unter anderem in [13] aufgezeigt. Um dagegen zu agieren, werden in hier verschiedene Punkte aufgezeigt, die für ein Testszenario gewählt werden sollten.

Diese können wir folgt definiert werden:

- Zu testende Knoten sollen zufällig verteilt sein
- Es existiert keine Gleichverteilung der Knoten innerhalb des Netzes
- Ankerpunkte sind nur in einigen Bereichen vorhanden, aber nicht verteilt
- Vorzugsweise sind Ankerknoten am Rande des Netzes angesiedelt

Der Autor des Papers [13] übt somit Kritik an den Simulationen anderer Verfahren aus und gibt gleichzeitig Richtlinien für das bessere Testen für die Entwicklung zukünftiger Verfahren.

B. Vergleich der vorgestellten Verfahren

Die vorgestellten Verfahren zeigen, in chronologischer Reihenfolge, eine Tendenz dazu virtuelle Koordinaten anhand von Hop Distanzen in einem Vektor zu speichern. Einige Verfahren arbeiten hierbei mit einer festgelegten Anzahl an Ankern, während andere diese Anzahl variieren lassen (z.B. BVR).

Das Ziel der neueren Verfahren ist es die Energieffizenz und die Routability zu steigern, sowie die Korrekturen während des aktiven Routing gering zu halten.

Name	# Anker	Vorteile	Nachteile	Jahr
VC für connectivity- based Protokolle	Keine	höhere Lokalisierung, Map- ping	Zuweisung kann fehlschla- gen, kein geometric greedy routing möglich	2004
Vcap	3	Koordinaten weit auseinan- der gesetzt, geografisches Routing möglich	bei Ausfall muss Protokoll neu gestartet werden	2005
VDCR	variabel	bei vielen Sensorknoten sehr effektiv im Routing	viel Aufwand durch viele Modi, um Probleme zu um- gehen	2009
BVR	variabel (Bea- con Knoten)	Korrekturen mit wenig Auf- wand möglich	Fluten des Netzwerkes im schlimmsten Fall (ineffek- tiv)	2005
LCR	4	Inkonsistenzen werden leicht umgangen	Inkonsistenzen sind trotz- dem vorhanden und müssen umgangen werden	2004
CSR	3	Verhinderung lokaler Mini- ma, keine Fehlschläge, ex- terne Anker können inte- griert werden	Energieeffizienz und Routa- bility verbesserungsfähig	2009
DVCR	pro CT Strang 2, bei Zwei- gen +1	100% Routability, ENS bringt Verbesserung, einfaches Schema	keine Bekannten	2011

Tabelle I VERGLEICH DER VERFAHREN

Das CSR Verfahren zeigt hierbei einen ersten Ansatz, dass durch ein effektives Setzen von virtuellen Koordinaten das Problem während dem Routing verringert werden kann.

Das DVCR Verfahren vereinfacht dieses Prinzip, in dem es eine Richtung bestimmt, in die das Paket im Routing später versendet werden kann.

Somit ist zu erkennen, dass die neueren Verfahren Ansätze und Konzepte aus früheren Verfahren übernehmen und verbessern. Ein direkter Vergleich ist dabei Tabelle 1 zu entnehmen.

V. ZUSAMMENFASSUNG UND AUSBLICK

Die virtuellen Koordinatenräume können auf verschiedene Art und Weise genutzt werden. In dieser Arbeit wurde sich auf die Verfahren konzentriert, die über Hop Distanzen die Abstände wählen. Es wurde gezeigt, dass es trotz gut gewählter Ankerpunkte immer noch zu Fehleranfälligkeit kommt. Diese Fehler haben meist den Ursprung im Fehlen einer Richtung, in die das Paket geschickt werden soll. Oftmals mussten zusätzliche Modi ausgeführt werden, um Pakete weiterzuleiten, die in Sackgassen geraten sind.

Diesem Problem wirken die gerichteten virtuellen Koordinaten entgegen. Dieses neue Konzept wurde durch eine Evaluation für Ankerpunkte erweitert und die Ergebnisse der Simulation zeigten weniger Fehleranfälligkeit. Zusätzlich wurden noch Topologie Maps eingeführt, die das Routen auf virtuellen Koordinaten beschleunigen.

Verbesserungen der Topologie Maps, der gerichteten Koordinaten und der Evaluation der Ankerpunkte ist in Zukunft zu erwarten, besonders der Autor *Dhanalapa* ist dabei zu nennen. Interessant ist auch, wie sich die gerichteten Koordinaten in verschiedenen Szenarien bewähren. Zur Zeit ist nur das Szenario eines CT bekannt, weitere wurden noch nicht untersucht. Hierbei können Verbesserungen im Vergleich zu anderen Verfahren auftreten, oder aber auch Verschlechterungen. Genauso kann die Kombination aus den vorgestellten Verfahren mit dem DVCS zu einer weiteren Verbesserung führen.

LITERATUR

 Geographic routing without location information. In Proceedings of the 9th annual international conference on Mobile computing and networking, pages 96–108. ACM, 2003. I

- [2] Anchor Selection and Topology Preserving Maps in WSNs A Directional Virtual Coordinate Based Approach. cnrl.colostate.edu, 2011.
- [3] J.N. Al-Karaki and A.E. Kamal. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6–28, 2004.
- [4] Qing Cao and Tarek Abdelzaher. A Scalable Logical Coordinates Framework for Routing in Wireless Sensor Networks . Real-Time Systems, (Rtss), 2004. II-A, III-E
- [5] A. Caruso, S. Chessa, S. De, and A. Urpi. GPS free coordinate assignment and routing in wireless sensor networks. In *INFOCOM 2005*. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, volume 1, pages 150–160. IEEE, 2005. III-B
- [6] D.C. Dhanapala and A.P. Jayasumana. CSR: Convex Subspace Routing protocol for wireless sensor networks. In *Local Computer Networks*, 2009. LCN 2009. IEEE 34th Conference on, number October, pages 101–108. IEEE, 2009. II-A, 2, III-F, III-F
- [7] D.C. Dhanapala and A.P. Jayasumana. Dimension Reduction of Virtual Coordinate Systems in Wireless Sensor Networks. *cnrl.colostate.edu*, pages 1–6, 2010. III-G
- [8] D.C. Dhanapala and A.P. Jayasumana. Topology preserving maps from virtual coordinates for wireless sensor networks. In *Local Computer Networks (LCN)*, 2010 IEEE 35th Conference on, pages 136–143. IEEE, 2010. II-B. 1
- [9] D.C. Dhanapala and A.P. Jayasumana. Directional Virtual Coordinate Systems for Wireless Sensor Networks. In *Communications (ICC)*, 2011 IEEE International Conference on, number June, pages 1–6. IEEE, 2011. (document), I, III-G, III-G, 3, 4, 5, 6
- [10] Rodrigo Fonseca, Sylvia Ratnasamy, Jerry Zhao, and CT Ee. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. *Proceedings of the 2nd*, pages 329–342, 2005. II-A, III-D
- [11] Chih-Cheng Hsu, Cheng-Fu Chou, Kate Ching-Ju Lin, and Chien-Chun Hung. Virtual Domain and Coordinate Routing in Wireless Sensor Networks. 2007 IEEE Symposium on Computers and Communications, pages 1–6, 2009. II-A, III-C
- [12] Brad Karp and H.T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In Proceedings of the 6th annual international conference on Mobile computing and networking, pages 243–254. ACM, 2000. III-A
- [13] A. Kroeller, S.P. Fekete, Carsten Buschmann, Stefan Fischer, and Dennis Pfisterer. Koordinatenfreies Lokationsbewusstsein (Localization without Coordinates). Arxiv preprint cs/0502069, 47:70–78, 2005. II-A, IV-A, IV-A
- [14] A. Kroeller, S.P. Fekete, Carsten Buschmann, Stefan Fischer, and Dennis Pfisterer. Universität Stuttgart Fakultät Informatik, Elektrotechnik und Informationstechnik Gruppierung von Sensorknoten auf Basis von Sensordaten und Konnektivität. Elektrotechnik Und Informationstechnik, (2487), 2006. II-A
- [15] Thomas Moscibroda, R. O'Dell, M. Wattenhofer, and Roger Wattenhofer. Virtual coordinates for ad hoc and sensor networks. In *Proceedings of the 2004 joint workshop on Foundations of mobile computing*, pages 8–16. ACM, 2004. III-A
- [16] D. Niculescu and B. Nath. Ad hoc positioning system (aps). In Global Telecommunications Conference, 2001. GLOBECOM'01. IEEE, volume 5, pages 2926–2931. Ieee, 2001. II-A
- [17] A Pflaum. Sensornetzwerke. *Internet der Dinge*, 49(0), 2007. I
- [18] C.S.J. Rabaey and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In USENIX technical annual conference, 2002. II-A
- [19] A. Savvides, H. Park, and M.B. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. In Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, pages 112–121. ACM, 2002. II-A
- [20] Robert Szewczyk, Eric Osterweil, Joseph Polastre, Michael Hamilton, Alan Mainwaring, and Deborah Estrin. Habitat monitoring with sensor networks. Commun. ACM, 47:34–40, June 2004. I

Attack and Counter Measures in P2P based Streaming Systems

Richard Gross

Abstract—Peer-to-Peer (P2P) based streaming solutions have gained significant popularity in China. Commercial streaming systems, such as PPLive and PPStream, have over two million daily customers. Furthermore, various other systems have already been developed and tested. Some based on the most popular P2P protocol, BitTorrent, and others, using an original core. However, with this success comes a need for security. Streaming systems based on P2P pose security risks, such as pollution, Sybill, eclipse or collusion attacks, that need to be addressed to ensure commercial viability and quality of user experiences. This survey will focus on six different streaming systems and the security threats they are exposed to.

I. Introduction

Peer-to-Peer (P2P) networking is a technique to distribute load among peers. With the HTTP protocol all upload cost is placed on the server that hosts files for interested clients. A client that has partly or completely finished the download will never supply content to other clients. In P2P systems, on the other hand, every client also acts a server and provides the data parts he has himself already downloaded to other interested client. Thereby the uplink load on the original proprietor of the data is reduced. At the end of this process, every interested peer will have a duplicate of the complete data. This high redundancy means that even when some peers are unavailable it does not affect the availability of the data. Theoretically information is no longer lost. In summary, P2P system offer benefits like "better load balancing, dynamic information repositories, redundancy, fault tolerance" [14], do not not necessarily have a single point of failure and can be very cost effective for the content distributor. Content providers can use P2P to increase the availability of their data without having to massively increase their own server upload capacity. P2P has become ever more important in the modern internet, evidenced by the popularity of BitTorrent, the leading P2P protocol. BitTorrent related traffic is estimated to make up anywhere between 18%-35% of all Internet traffic [2, 3] and companies are increasingly using P2P techniques to distribute their content.

In P2P based streaming systems peers regularly communicate with each other and exchange control or content messages. On one hand the system would not work without these messages. On the other hand the messages should not be trusted implicitly either. An example for this is when malicious peers share video parts that are not correct, e.g. full of noise. A defensive mechanism must be in place to ensure that received video parts are correct and not part of such a *pollution* attack. There are many other attacks which undermine trust properties and facilitate the denial of service

(DoS) for a user. To clarify, when a service is in some way rendered unavailable, we speak of a DoS attack. The focus of this seminar is to find DoS attacks like pollution, analyse them and propose defences.

The rest of this paper is organized as follows. In the next section we will describe P2P systems that are used for audio or video streaming. The following third section will summarize possible DoS attacks on these P2P streaming system. Section IV proposes defences to these attacks. The fifth section proposes which defence mechanisms to utilise and the subsequent final section contains the conclusion.

II. P2P STREAMING SYSTEMS

In this section we will describe six different P2P streaming systems. In our summary we focus on certain aspects of these systems: design goal (what was the goal of the system), join procedure (how peers join the network and if the network can be joined by anyone, open membership, or if there are restrictions), peer identity (how the peer identity is determined), peer communication (how peers communicate with each other), peer structure (structured or unstructured or if are there specialised peers), content discovery (the way by which peers find content), video scheduling (the scheme by which peers decide which video parts to download), video transmission (how the pieces are transmitted, e.g. unicast or multicast) and video verification (how the correctness of the pieces is verified). Every system description is accompanied by a brief table that summarises all these properties. These systems are further condensed by the table on page 9.

A. Tribler

Tribler is an open-source software project that extends the BitTorrent protocol through client-side modifications but remains compatible. The distinguishing feature is that Tribler, as detailed by Pouwelse et al. [25] in their paper from 2006, exploits social phenomena to increase usability and performance. This original version did not yet support streaming and was only intended for file sharing. It it can be assumed that the successful SwarmPlayer field study (a modified Tribler client, SwarmPlayer, supported live video streaming) by Mol et al. [22] convinced the Tribler team to include live streaming into the Tribler core. More recent papers by Zeilemaker et al. [33], [34] seem to concur with this assumption. Therefore we base our assumptions on the previously mentioned papers [22], [25], [33], [34].

Tribler peers contact a set of pre-known superpeers to obtain an initial list of peers in the system. Gradually the Tribler client uses query messages to extend the number of peers with taste buddies (similar content interest) and random peers (no recognised interest) it knows. Information such as "social relations, altruism levels, peer uptimes" [25] and what content a node finds interesting is continuously exchanged between peers with the help of the BuddyCast algorithm. Peers that rate content similar to each other are thus more likely to have each other in their list of taste buddies. In Tribler taste buddies server an important purpose for content discovery and content acquisition. Peers can use the similarity between their preferred content and the preferred content of other peers to calculate and suggest recommendations. This marks a stark difference to BitTorrent, where content discovery always requires user initiation. Taste buddies also improve content acquisition through collaborative downloading. In Tribler peers can assist each other by downloading and then sharing a piece of content that the other wants. This form of cooperation can even be used if a peer does not even desire the content. Consider the case that peer A and B desires chunks from peer C, whereas peer D does not want this specific content. Peer C only has enough upstream bandwidth to transmit a chunk to a single peer. Peer D has adequate upload and download bandwidth for all three peers A, B and C. Using Tribler peer C can transmit the chunk to peer D, which in turn transmits the chunk to both B and C, effectively a multicast operation. The 2Fast protocol, which supports this type of cooperation, is further detailed in the research paper by Garbacki et al. [7]. Such strong cooperation requires trust, which in turn requires stable identities. To this end Tribler introduces "permanent, unique, and secure peer identifiers (PermIDs), which are [...] public keys [...] and which are exchanged by e-mail" [25]. A random number-based challenge-response mechanism then makes sure the peers are what they appear to be.

The Tribler live streaming extensions deploy a sliding window to determine which pieces to share and which are out-dated. Majority voting (different playback positions are compared and the most popular position is considered to be correct) determines what the correct current playback position is. Instead of hashing, every piece is signed by the injector and supplied with an absolute 64-bit sequence number and time stamp. Chunk selection is done 'rarest first' (the chunk with the lowest availability) in the sliding window. Finally Tribler utilises a variant of super peers. These peers are trusted and always supplied enough bandwidth by the injector. These peers take load of the injector and increase piece availability. To prevent malicious behaviour the super peers identities are not publicly known.

B. CoolStreaming/DONet

CoolStreaming is the public Internet-based implementation of the DONet protocol. DONet is the technically sound, while CoolStreaming is the commercially more appealing name, but they essentially describe the same technology. DONet is a

Data-driven **O**verlay **Net**work developed by Zhang et al. [35] that relies on P2P to distribute content.

The protocol does not maintain an explicit structure of the peer nodes. Every node maintains a membership cache (mCache), containing a partial list of the identifiers for the active nodes in DONet. This cache is used for operations like selecting streaming partners and gets periodically updated through membership messages. A membership message contains the sequence number of the message, the unique node identifier (typically the network address), the number of streaming partners and the remaining time the message is valid for. These messages are send to other peers using the Scalable Gossip Membership Protocol (SCAM) [6]. Upon joining every new node first contacts the origin node of the stream, which redirects it to another node, the deputy node. This randomly selected node in turn supplies the new node with a list of partner candidates for video streaming. Nodes circumvent partner failures by periodically establishing new partnerships, randomly selected from the mCache. This way nodes can also find partners which have more partners and segments available.

Every video stream is divided into segments of uniform length, typically one second long, and the availability of segments is announced via a buffer map (BM), typically 120 segments long. Like a sliding window, the sequence number of the first segment in the BM is announced through a two byte value. Each node continuously shares its BM with its streaming partners. Using the BM of its partner every node downloads the rarest segment first from the supplier with the highest bandwidth and enough available time.

C. PPLive

PPLive is a proprietary P2P video streaming system. All analysis results by Hei et al. [12] were achieved with the help of an active crawler, a passive sniffer and the small amount of publicly available information. Additional information is derived from a paper by Horvath et al [13]. Since there is no knowledge of the specific inner workings of the system available it can only be assumed that the following description of how PPLive works is close to the truth. When joining the service, a peer first retrieves a list of all the 320+ channels from the central server. This list also describes which tracker is responsible for which channel. Peers then download and share information about the chosen channel with the tracker server. Perhaps in an effort to reduce load on the tracker, he does not have to communicate every peer in a channel to newcomers. It falls within the responsibility of peers to talk to their streaming partners, obtain additional lists and aggregate them to their existing list. "PPLive peers greedily contact other peers, at an almost constant rate. After one hour, up to 40.000 peers can be contacted by a single node." [13] This behaviour is perhaps due to the very short peer lifetime.

Chunk distribution is accomplished using a sliding window. "Buffer map" messages between peers communicate which chunks the peer has available and currently buffered. It is unclear if rare or soon-to-be-played chunks are prioritised

or how complex the chunk selection algorithm is. It has been observed, however, that PPLive favours local peers for streaming. Especially those with high bandwidth are prioritised. This suggests that peer regularly gossip their remaining upload capacity. The amount that a peer has uploaded does not affect his download rate, which suggests that PPLive does not have a policy similar to *Tit-for-Tat* (BitTorrent).

D. SopCast

SopCast is a free closed-source P2P streaming system, born as a student project at Fundan University in China. The information presented in this chapter is aggregated from various research papers ([5], [13], [21], [27]) and usually derived through modified clients, packet sniffing or similar, with no knowledge of the specific inner workings of the system.

The SopCast protocol operates similar to BitTorrent with every channel forming a mesh-based network of peers. When a node first joins the network it contacts a central server, located in China. From there it retrieves an updated channel list. In SopCast every peer can broadcast its own channel. After the peer has selected a channel he contacts the corresponding trackers and retrieves a current peer list. Research suggests that the joining peer then proceeds by contacting other peers in a seemingly random fashion. The location of these peers is not considered. The protocol then seems to select the peers with the highest bandwidth. The information about the upload capacity and chunk availability of a peer is probably communicated to other nodes using a gossip-based protocol. Peers are periodically contacting other peers to form new streaming partnerships but the SopCast protocol limits this rate after the initial bootstrapping phase. SopCast uses almost exclusively the UDP protocol for transmission, presumably because it provides a lower delay compared to the TCP protocol. The high packet loss rate leads to a substantial amount of overhead, in the form of control or signalling packets. About 60% of the packets are signalling packets versus 40% of actual video data packets. One reason for this is that, unlike the mostly TCPbased PPLive, SopCast has to send application-layer package reception acknowledgements.

The specifics of video scheduling in SopCast are not clear. Research results show that there is no tit-for-tat policy, i.e. a peers upload rate does not determine its download rate. There does, however, seem to be a limit on the download rate, as it is on average the same for every peer in the network.

E. TrustStream

TrustStream by Yin et al. [32] promises to be a system with "unprecedented security, scalability, heterogeneity". The researchers argue that their system merges the best properties of CDNs (security and accommodating QoS) and P2P media streaming (scalability, inexpensive cost of deployment) and actualises copyright management and access control for commercial content providers. This new peer-server-peer (PSP)

structure provides these features through a two-layer streaming architecture. A very low bit rate "base layer" and a much higher bit rate "enhancement layer". The base layer provides the core video. The optional enhancement layer mainly improves the quality of the stream and can only be decoded if the base layer is available. The base layer distribution is accomplished through a scalable hierarchical multicast structure while the enhancement layer employs mesh-based P2P unicast system that adopts gossip-based P2P membership management. According to Yin et al. TrustStream been implemented in real network and broadcasted several nationwide popular live video programs all over China and has been adopted by ChinaCache, the largest CDN provider in China.

Members of the base layer are ad hoc assigned to different layers of the streaming tree according to their performance (bandwidth, CPU etc.) and round-trip delay time (RTT). The first layer contains carefully chosen CDN-featured fixed nodes. Members in each layer are further grouped into clusters. Each cluster has a cluster leader that multicasts data to all the members of its cluster. The cluster leader should have the maximum local performance and minimum average distance (RTT) to other members in the cluster. A node joins the base layer by contacting the main server which will conduct the certification authority (CA) verification. If successful the server will then sign a time-limited label for the identity of the new peer node and send a list of recommended cluster leaders. The base layer stream is encrypted by a session key (SK, shared by all group members and periodically refreshed) and cluster key (CK, shared by each cluster member and periodically refreshed by the cluster leader). The CK is always updated when cluster members change to ensure the security of the stream, i.e. that no unauthorised node can access the stream. TrustStream embeds key messages in the host video signal to further increase security and reduce bandwidth consumption.

The **enhancement layer** is organised in a mesh-based fashion. Every node forms partnerships with a random set of other peer nodes. Through a gossip protocol (Lin et al. [20]) it can then obtain a partial view of group membership. Every node periodically re-evaluates its partners and aims to form optimal bandwidth partnerships. Synchronising enhancement and base stream is accomplished through pre-fetching the base layer in the buffer and waiting for the enhancement layer. Like the base stream the source for the enhancement stream is a CDN.

F. DagStream

DagStream is a novel streaming system, developed by Liang et al. [19], which aims to solve general P2P challenges such as peer unreliability, bandwidth heterogeneity and network efficiency.

In unstructured mesh-based P2P streaming systems, peers select neighbours based on their locally perceived performance. Due to this selection scheme, peers with high resource availability become single points of failure, that act as conjunctions between different network clusters. Should any

one of them fail, the network would be split into disjunct parts, rendering the system broken. Liang et al. suggest self-organising the peers into a directed acyclic graph (DAG) where each peer has at least k parents. As long as less than k peers fail at the same time the node will stay connected to the network. The source of the video stream is the root of the DAG. Each peer can only stream data from its parents and can only provide data to its children. Furthermore the DAG is locality aware. Peers use the network more efficient than mesh-based systems, since they stream content as often as possible from nearby peers.

DagStream uses the locality first, bandwidth second property, selecting nearby peers by choosing those with smaller delay. Additionally peers also try to find parents with a small level (the number of peers between the parent and the content source) to minimise playback delay.

Peers join DagStream and find their parents using the separate service, *RandPeer* [18]. Every peer, even the source, periodically registers with this distributed membership service network. When a peer, such as a newly joined one, needs to locate a parent, it sends a lookup request to the RandPeer service and receives membership information of another registered peer. If this peer has less children than a threshold *C max* it will add the lookup peer as a child. Since peers are clustered based on their stated QoS characteristics they can easily lookup locally close peers. Peers will periodically probe new peers and, based on the probing results (peer delay, level in the DAG, sufficient child slots), select them as parents. Every peer strives to maintain equal or greater than *k* parents and to this end either accepts any new parents (less than k parents) or drops less desirable parents (more than k parents).

An already joined node can discover new parents through their own two-hop neighbour list (parents and children periodically send heartbeats and inform each other of their neighbours) or parent suggestion (peers suggest themselves as parents to other low-delay peers). The DAG is kept loop-free because peers only accept parents with a level, i.e. the hop distance in the DAG to the source, smaller than their own. Loops can still happen though, if multiple peers select parents at the same time. Thus the system needs to actively detect loops. A loop will increase the level of a parent node after every refresh period. Thus the child node needs to check if the level of their parents increases by a suspicious margin over time.

In DagStream each parents notifies their children about the data blocks they have available. The children then select the best parent based on their data availability and bandwidth availability. The specifics are not clear though since Liang et al. [19] focus their paper on the data scheduling and on building the underlying DAG in a distributed fashion.

III. ATTACKS ON P2P STREAMING SYSTEMS

All streaming systems require delay-free video and audio playback to satisfy the most basic customer expectations.

Thus, any attempt by a potential attacker to diminish or completely deny the video service needs to be thoroughly analysed and, in later chapters, a solution must be proposed.

Attacks on P2P systems can generally be divided into attacks on the data or the control plane. If a malicious peer drops, corrupts, delays, duplicates or forges content data, it is considered an *attack on the data plane*. If the malicious peer gives wrong routing or control information to another node, accepts to many downstream peers, connects to multiple upstream peers, advertises fake data availability, it is considered an *attack on the control plane*.

Using the survey by Gheorge et al. [8] as a base we identify six common attacks in P2P streaming systems that we will explore further: Sybil, whitewashing, flooding, membership, omission and pollution attacks. It is worth noting that any of these schemes can be combined with the Sybil attack, the forging of multiple identities, to make them much more powerful and dangerous. In cases were independent peers work together to make an attack much more dangerous, we speak of a collusion attack. One general example for this is when an "attacker compromises a set of nodes to perform a coordinated attack to the system" [11]. Another example is when "peers work together to misrepresent information" [29]. The goal of the attacker and method of attack is not clear in this case due to collusion being considered a clustering term for the flooding and the omission attack.

All these attacks and their colluding behaviour are summarised in the table on page 10.

A. Sybil

A Sybil attack is when at minimum a single entity assumes multiple identities in the same system. The attack misuses the peer identities of open P2P systems, where the identity is only tied to the network address, e.g. DONet. Internet addresses can easily be forged, however. The abstract goal of the attacker is to significantly increase his influence in the system. Usually the attacker uses the increased influence to make attacks like flooding, omission, pollution etc. more effective. Special versions of the Sybil attack which have the goal to partition the P2P system are discussed in the 'Eclipse' and 'Black Hole' paragraphs.

B. Whitewashing

Whitewashing is an attack that can be launched on systems that tie a peers rating to their identity. An example for this is the Tit-for-Tat scheme that BitTorrent and thereby Tribler employs. This scheme ties a peers download rate to its upload rate. An exception to this rule is the launch phase, where peers do not have any pieces to share and can download freely. Malicious attackers can thus download as much as possible until their rate starts to suffer from lack of uploading. Then they create a new identity and, washed clean of their bad rating, they can start downloading at high rates again.

C. Flooding

The **goal** of the attacker is to minimise the **reliability** of a target. This is accomplished through exhaustion of the resources of the victim by sending a large quantity of messages. The victim is thus unable to comply with any other messages until all "flooded" messages have been processed. The genuineness of these messages does not even matter as long as the recipient is occupied by simply receiving it. Usually this attack is launched by members of a P2P streaming system against a victim that is not part of the same.

The attack can be **launched** by using the epidemic nature of gossip control messages. Gossip protocols are used in a large variety of P2P systems to propagate information in a decentralised manner. An attacker in a DONet, PPLive, SopCast etc. system could send out a large amount of messages which all assert that a target has popular content and a large amount of bandwidth available. This message will then be spread to other peers which in turn will forward this message. The attacker could also have modified the lifetime of the message, leading to an indefinite false information spread. Due to the content being popular, all interested peers, that have received the fraudulent gossip message, will contact the target at similar times, completely overwhelming the available resources.

An example for flooding, without malicious intent, is when content on a webserver suddenly increases in popularity. We call this non-malicious flooding, a flash crowd. Flash crowds show us that a flood of requests cannot simply be ignored. We have to distinguish between detrimental flooding attacks and flash crowds and only serve the latter.

D. Omission

The omission attack targets the reliability of the P2P system and hampers the communication between nodes. The goal of such an attack is to reduce the QoS for a subset of peers in the P2P system. An omission attack is accomplished by not forwarding control information (peer management information, piece availability messages) to other legitimate peers.

In systems which employ gossiping, DONet, PPLive, Sop-Cast etc., it can be very damaging. In DONet peers periodically establishing connections with other peers they have stored in their *mCache* to improve their partnerships and stay resilient to churn. The *mCache* is filled with peer information obtained from partners via gossip messages. Suppose malicious peers omit such messages that they normally would forward. High churn and unreliable connections could gradually lead to a depleting mCache and no new established partnerships, in the end leaving only the malicious, omitting peers. The more peers omit information the higher the probability of this scenario but even fewer malicious peers can significantly reduce the quality of the partnerships and thus, the streaming quality.

E. Pollution

P2P streaming systems are strongly affected by pollution attacks. In a pollution attack the aggressor mixes his soiled data with the correct stream data, reducing the video quality as well as destroying data integrity and authenticity.

The exploited vulnerability are the buffer maps that protocols, such as DONet, utilise to announce piece availability. Malicious peers can announce to other peers that they have (almost) all pieces available. In reality the pieces are full of arbitrary data. One could also suppose that an attacker might inject his own advertisements into the video stream, replacing normal content or the advertisements of the original video proprietor. Peers have no way of knowing if the announced pieces are real until they have downloaded them. Should they not discover the fraud the pieces might even be forwarded by honest peers, degrading the QoS for even more honest peers [3].

Haizhou et al. detail how a pollution attack can easily pollute PPLive [9]. Their measurement study shows that a *sin-gle* polluter with a 10/100Mbps Ethernet network access can easily disrupt the system and significantly reduce the quality of service. During a time frame of about 30 minutes the number of regular viewers decreases rapidly from approximately 3000 within minutes of launching the attack. After 15 minutes the viewer count falls to 1500 and decreases much slower now. 45 minutes after launching the attack there are about 1000 viewers left. After 1 hour and 30 minutes there amount of viewers reaches the lowest point at around 550. Haizhou et al. surmise that the viewers do not drop below this point because the leavers are balanced by the new joiners and other stubborn ones who rejoin.

F. Membership and Eclipse

Membership attacks target the "themembership protocol or the way nodes are admitted into the overlay" [8]. A specialised version of this is the eclipse attack. "If an attacker controls a large fraction of the neighbors of correct nodes, it can 'eclipse' correct nodes and prevent correct overlay operation." [28] Eclipse attacks practically affect any P2P system that uses an open, decentralised overlay network such as DONet or DagStream. In both systems nodes receive membership information from their peers. A malicious attacker can gossip information to other peers that leads them to unjustifiably exclude or include peers based on the attacker bidding. This biased information is then passed on by the honest node to their peers in turn. After a while malicious attackers can attract and surround a large number of honest nodes, 'eclipsing' them from the other honest peers in the network.

G. Neighbourhood selection

When an assailant controls the mesh overlay formation and maintenance we speak of a neighbourhood selection attack. In PPLive newly joined peers aggregate information from their streaming partners to determine other potential partners. Colluding peers could refer every new joining peer to other colluding peers (the eclipse attack, where honest nodes are surrounded by malicious ones).

Seibert et al. [26] show that malicious peer nodes can dominate joining honest nodes by only referring other malicious nodes as neighbours. Attacks that can result after this malicious neighbour referring is selective data forwarding,

cheating, traffic analysis, overlay partitioning, or flooding attacks targets that might not even be part of the network. Some of them are detectable by honest nodes, such as selective data forwarding, while other attacks, like traffic analysis, do not have immediate observable results. By using a large number of malicious peers, achievable through a Sybil attack, the attacker could even infiltrate the main membership server and disrupt the whole P2P system before being noticed.

IV. DEFENSES AGAINST ATTACKS ON P2P STREAMING SYSTEMS

The distributed nature of P2P streaming systems generates many new attack schemes compared to client/server systems. This section describes the popular defences that have been devised for the security problems detailed in the previous section.

A. Defences for Sybil-based attacks on P2P Streaming Systems

There are several proposed solutions to the Sybil attack. Trusted certificates, resource testing, recurring costs, trusted devices and observation are summarised by Levine et al. [17]. Some researchers even believe there is no solution. Douceur [4] argues that practical P2P systems can only prevent Sybil attacks by employing a *trusted central agency* that *certifies identities*. The most common form is *resource testing* and requires known resource limitations for either communication, storage or computation.

'The only direct means by which two entities can convince a third entity that they are distinct is by performing some task that a single entity could not.' [4].

In a P2P systems, peers would periodically request their connected peers to solve certain challenges. To prevent an attacker from simply solving the challenges to his multiple identities one-after-another, they have to be issued at exactly the same time to every peer in the network. Typical identities in P2P systems, however, don't have nearly identical resource constraints and the global coordination of puzzles is not possible without a trusted central party. This effectively makes completely decentralized P2P systems always susceptible to Sybil attacks. A conclusion most researches agree on, based on the Sybil survey by Levine et al..

A variation of resource testing is the concept of *recurring cost* and fees for the attacker. Identities are periodically revalidated. An example of this are Turing Tests that can only be solved by humans and not automatically by a machine. CAPTCHAs [1] being the current prime example of this approach. Haribabu et al. [10] propose a solution that exploits the common behaviour characteristics of Sybil attackers. A neural network is run on a central server and is attempting to deduct which identities are part of a Sybil scheme. The neural input data is supplied by so called *ultrapeers*. Ultrapeers aggregate information about their leaf peers and improve scalability of the system. To reduce computation overhead, only peers which are suspected by the neural network are send CAPTCHAs. In

a small scale simulation with 110 nodes and ten ultrapeers this approach has been shown to effectively identify Sybil peers, even if they pretend to be ultrapeers. The researchers assume honest peers will always be able to solve this query. However CAPTCHAs suffer some usability issues, described by Yan et al. [31] and it is also worth noting that while CAPTCHAs can be solved by machines with a usually below-human reliability, it is often cheaper to hire cheap labour that is paid \$0.5 per 1000 CAPTCHAs solved [23].

Certain networks could deploy trusted devices to ascertain credibility. Multiple devices could theoretically be acquired by the attacker but this should be prevented by the high cost per unit. Observation of the attacker is the last possible choice. In mobile networks the Sybil peers on the same device would always move together. Another form of observation is a reputation system. Here peers give each other ratings and distribute these ratings amongst themselves. Cheng et al. [2] argue that symmetric reputation systems like Google PageRank [24] are not sybil-proof. Flow-based asymmetric reputation systems rely on trusted nodes which originate and propagate trust values. Symmetric systems, by comparison, rely solely on the topology of their trust graph and largely ignore from where the values come.

Systems like TrustSystem on the other hand can make this attack too expensive for the attacker. TrustStream employs a CA and could couple every account with a subscription fee (making multiple accounts infeasible for the attacker).

B. Defences for Whitewashing-based attacks on P2P Streaming Systems

Defending against whitewashing is accomplished by making it unfavourable for attackers to renew their identity. Rating systems where the attacker first has to achieve a positive score before he earns important privileges are one idea. Theoretically this would be true for the BitTorrent Tit-for-Tat scheme. There exists a short window after identity renewal where peer download rates are unconstrained. New users rely on these startup download rates, however, making them a necessity of design. Similar to the Sybil attack, the only solution to whitewashing seems to be based on a central server.

C. Flooding

Flooding based attacks, that target a third party, are very hard to detect when triggered from within a P2P network. Suppose a rating system of some sort is in place were peers give and share peer ratings based on the information they provide. A detected malicious peer will not be able to trigger an attack. But suppose the malicious peer first behaves honest and then issues a singular attack. He will not be able to issue a second attack because his rating will decline after peers downrate it afterwards but whitewashing circumvents this defensive measure. The best way to defend against flooding attacks is then from the perspective of the target.

A paper by Jung et al. [15] argues that it is possible to differentiate between flash crowds and true flooding attacks based on their characteristic patterns. It is however reasonable to assume that quality streaming systems should have a

backup injector to accommodate for suddenly very popular content, the flash crowd scenario. The most practical solution seems to be to employ the aid of super peers the same way SwarmPlayer [22] does it and distribute an attacking flood of messages to be processed for validity at these super peers.

D. Defences for Membership-based attacks on P2P Streaming Systems

One solution to this attack are structured overlays such as distributed hash tables (DHT). Seibert et al. [26] describe another novel solution which works for unstructured meshbased networks. Their assumption is that an infiltrated mesh structure will be become significantly more organised. If the attack is significant enough, the structure will gain a dense center, full of malicious nodes. The more random a graph is the less links there are between neighbour nodes. After the attack, the connectivity of honest nodes through a malicious node will increase. By monitoring this metric honest nodes can effectively purge malicious nodes. Seibert et al. [26] show that their solution is very effective for stable overlays of up to one thousands nodes with at most 20% malicious nodes in the neighbourhood set. For greater overlays the number of malicious nodes is still significantly decreased but the convergence time also increases.

Another technique proposes to empty and refill routing tables periodically. Arbitrarily reforming the routing table makes it impossible for malicious nodes to constantly stay in the table of an honest peer node.

E. Defences for Omission-based attacks on P2P Streaming Systems

SecureStream [11] proposes a peer organisation into multiple unidirectional rings where peers can accuse and rebut accusations of malicious behaviour. The organisation into multiple rings minimises the chances that a peer would always communicate and only receive messages from malicious peers. Their experiments prove that even a high number of freeloading nodes cannot disrupt the system significantly. Periodic renewal of routing table is also a solution. Random neighbour selection would minimise the risk of too many malicious nodes in the routing table.

F. Defences for Pollution-based attacks on P2P Streaming Systems

Protection concepts against pollution include blacklisting, traffic encryption, hash verification and chunk signing [3]. Blacklisting peers is a simple concept which can easily be circumvented by the attacker with whitewashing. If the attacker can easily create new identities, blacklisting becomes ineffective. Traffic encryption could be employed for all messages to obfuscate and hide the messages from an attacker. However, this can at best be considered a delay tactic until the attacker has reverse engineered the protocol. Hash verification is not applicable in a scalable P2P streaming system. Any attacker can easily inject polluted content that nevertheless has the same hash value as the correct video chunk. Out of

all the solution chunk signing seems to be the most applicable one. There exist many different approaches for this scenario. To reduce processing requirements not every chunk has to be signed individually. By aggregating the chunks in different ways and signing these results it becomes even possible for PDAs to verify the validity of the received chunks.

V. DISCUSSION ON A SECURE P2P STREAMING SYSTEM

As seen in the previous sections attacks on P2P systems are interweaved with each other whereas defences usually only target a specific attack pattern. For example credit systems mitigate DoS attacks but are subjugated by whitewashing and Sybil attacks. The further rise to prominence of P2P streaming systems would ideally require a solution that handles all problems. In this section we propose a hypothetical system that achieves resilience against the most important security issues in P2P streaming systems.

This system could employ a peer rating scheme, were peers give and share trust scores with each other, that mitigates many attacks before they happen. PeerTrust, by Xiong et al. [30], is such a decentralised, reputation-based trust supporting framework based on a transaction-based feedback system. In this scheme peer continuously share feedback, about peers they have interacted with, with other peers. They receive similar feedback from their communication partners. To determine false feedback peers calculate the similarity between the feedback they have given a set of peers and another peer has given the same set of peers. Now consider that colluding peers tend to give good feedback inside their group and bad feedback outside their group. Hence their feedback similarity will be very low and their trust equally so. The researchers have shown that their systems works in different scenarios, even when the number of colluders reaches as high as 70%. Another possible choice would have been EigenTrust [16].

We assume that a mesh-based structure is much better adapted to the high churn of the internet environment and should be utilised by any modern P2P streaming system. To mitigate the Sybil problem our system would need a central certificate authority, similar to the one used by Trust-Stream, that issues asymmetric keys to peers. Whitewashing is made more difficult because new keys and identities are only generated through referrals, i.e. existing members of the network need to invite outsiders. This also makes it possible to identify and exclude nodes that refer a suspicious amount of misbehaving peers. Collusion attacks are mitigated by employing a system like PeerTrust, which has been shown to detect attacks even at high peer collusion levels. Practical tests will have to show if this scheme can also subdue flooding and omission attacks. We suggest that peers with a low trust level can download at reasonable speeds, no to discourage new users, whereas their overlay messages are not trusted. Malicious nodes need to first build a good reputation before they can influence the overlay network. Together with PeerTrust this helps us mitigating membership attacks such eclipse and neighbourhood selection attacks. PeerTrust also helps us combat pollution attacks, since polluted piece transactions receive bad feedback. Piece verification is accomplished through public key signing by the origin of the stream.

It remains to be tested if the proposed mechanism truly work in unison, keep the network free of misbehaving peers and do not, or only to a small degree, penalise honest peers.

VI. CONCLUSION

This survey has described P2P based streaming systems in general and six different systems in particular. Tribler, DONet/CoolStreaming, PPLive, SopCast, TrustStream and DagStream were further condensed into the table on page 9 according to traits such as design goal, join procedure, peer identity, peer communication, peer organisation, content discovery, video scheduling, video transmission and video verification. We then used these system traits to explain the following six attacks on P2P based streaming systems: Sybil, whitewashing, flooding, membership, omission and pollution. Furthermore we described defence schemes for these attacks, which in general constitute of various monitoring and auditing schemes. Attacks and their defences were condensed into the table on page 10. Finally we included a short discussion about a theoretical system, resistant to the attacks described in this paper.

In conclusion we think a growing P2P streaming popularity demands significant effort to be spend on security which has not been met by the free, open systems discussed in this survey. Certain closed systems (access to the system is controlled through a central, monetary authority) such as TrustStream, however, already have proficient security measures. It is our hope that open alternatives will take equally or more successful steps and through this security provide unprecedented quality of service for video on demand and live video streaming.

REFERENCES

- Louis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. CAPTCHA: Using Hard AI Problems for Security. Science, 2003. IV-A
- [2] Alice Cheng and Eric Friedman. Sybilproof Reputation Mechanisms. Industrial Engineering, 2005. IV-A
- [3] Prithula Dhungel, Xiaojun Hei, Keith W Ross, and Nitesh Saxena. The Pollution Attack in P2P Live Video Streaming: Measurement Results and Defenses. Science. III-E, IV-F
- [4] John R Douceur. The Sybil Attack. pages 1-6. IV-A
- [5] Benny Fallica, Yue Lu, Fernando Kuipers, Rob Kooij, and Piet Van Mieghem. On the Quality of Experience of SopCast. Measurement. II-D
- [6] Ayalvadi J Ganesh, Anne-marie Kermarrec, and Laurent Massoulie. Peer-to-Peer Membership Management for Gossip-Based Protocols. 52(2):139–149, 2003. II-B
- [7] P. Garbacki, a. Iosup, D. Epema, and M. van Steen. 2Fast: Collaborative Downloads in P2P Networks. Sixth IEEE International Conference on Peer-to-Peer Computing (P2P'06), pages 23–30, 2006. II-A
- [8] Gabriela; Gheorghe, Renato; Lo Cigno, and Alberto Montresor. Security and Privacy Issues in P2P Streaming Systems: A Survey. *Network*, (214412), 2009. III, III-F
- [9] Wang Haizhou, Chen Xingshu, and Wang Wenxian. A Measurement Study of Plluting a Large-Scale P2P IPTV System. III-E
- [10] K Haribabu, Dushyant Arora, and Bhavik Kothari. Detecting Sybils in Peer-to-Peer Overlays using Neural Networks and CAPTCHAs. *Technology*, pages 1–9. IV-A
- [11] M Haridasan and R Vanrenesse. SecureStream: An intrusion-tolerant protocol for live-streaming dissemination. *Computer Communications*, 31(3):563–575, February 2008. III, IV-E
- [12] Xiaojun; Hei, Chao; Liang, Jian; Linag, Yong; Liu, and K.W. Ross. A Measurement Study of a Large-Scale P2P IPTV System. IEEE Transactions on Multimedia, 9(8):1672–1687, December 2007. II-C

- [13] Akos Horvath, Dario Rossi, Paolo Veglia, Delia Ciullo, and Marco Mellia. Dissecting PPLive , SopCast , TVAnts. Work. II-C, II-D
- [14] An Information-sharing and B Andrew. P2P Networking: An Information-Sharing Alternative. Networks, (July):31–38, 2001. I
- [15] Jaeyeon Jung, Florham Park, and Michael Rabinovich. Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. *Traffic*. IV-C
- [16] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. ReCALL, 2003. V
- [17] Brian Neil Levine, Clay Shields, and N Boris Margolin. A Survey of Solutions to the Sybil Attack. World. IV-A
- [18] J. Liang and K. Nahrstedt. RandPeer: Membership Management for QoS Sensitive Peer-to-Peer Applications. Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications, pages 1–10, 2006. II-F
- [19] Jin; Liang and Klara Nahrstedt. DagStream: locality aware and failure resilient peer-to-peer streaming. *Proceedings of SPIE*, 6071:60710L– 60710L–15, 2006. II-F
- [20] Meng-jang; Lin and Keith Marzullo. Directional Gossip: Gossip in a Wide Area Network. Architecture, pages 1–14. II-E
- [21] Yue Lu, Benny Fallica, Fernando A Kuipers, and Robert E Kooij. Assessing the Quality of Experience of SopCast Piet Van Mieghem. Electrical Engineering, 4(1), 2009. II-D
- [22] J. J. D. Mol, A Bakker, J. a. Pouwelse, D. H. J. Epema, and H. J. Sips. The Design and Deployment of a BitTorrent Live Video Streaming Solution. 2009 11th IEEE International Symposium on Multimedia, pages 342–349, 2009. II-A, IV-C
- [23] Marti Motoyama, Kirill Levchenko, Chris Kanich, Damon Mccoy, Geoffrey M Voelker, and Stefan Savage. Re: CAPTCHAs – Understanding CAPTCHA-Solving Services in an Economic Context. Forum American Bar Association. IV-A
- [24] The Pagerank, Citation Ranking, and Bringing Order. The PageRank Citation Ranking: Bringing Order to the Web. World Wide Web Internet And Web Information Systems, pages 1–17, 1998. IV-A
- [25] J A Pouwelse, P Garbacki, J Wang, and A Bakker. Tribler: A social-based peer-to-peer system. Computer, pages 1–11, 2006. II-A
- [26] Jeff Seibert, Sonia Fahmy, and Cristina Nita-rotaru. 'Won't You Be My Neighbor?' Neighbor Selection Attacks in Mesh-based Peer-to-Peer Streaming. Computer, 2008. III-G, IV-D
- [27] Alexandro Sentinelli, Gustavo Marfia, Mario Gerla, and Leonard Kleinrock. Will IPTV Ride the Peer-to-Peer Stream? *IEEE Communications Magazine*, (June):86–92, 2007. II-D
- [28] Atul Singh, Miguel Castro, Peter Druschel, and Antony Rowstron. Defending against Eclipse attacks on overlay networks. Auditing. III-F
- [29] Xiao Su, San Jose, Suchreet K Dhaliwal, and Cisco Systems. Incentive Mechanisms in P2P Media Streaming Systems. *Ieee Internet Computing*, 2010. III
- [30] Li; Xiong and Ling Liu. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. V
- [31] Jeff Yan, Ahmad Salah, and El Ahmad. Usability of CAPTCHAs Or usability issues in CAPTCHA design. Framework, 2008. IV-A
- [32] Hao Yin, Chuang Lin, Qian Zhang, Zhijia Chen, and Dapeng Wu. TrustStream: A Secure and Scalable Architecture for Large-Scale Internet Media Streaming. *IEEE Transactions on Circuits and Systems*, 18(12):1692–1702, 2008. II-E
- [33] Niels Zeilemaker, Mihai Capota A, and Johan Pouwelse. Tribler: P2P Media Search and Sharing. *Channels*, pages 739–742, 2011. II-A
- [34] Niels Zeilemaker, Mihai Capot, Arno Bakker, and Johan Pouwelse. Tribler: Search and Stream. *Processing*, pages 2–3, 2011. II-A
- [35] Xinyan; Zhang, Jiangchuan; Liu, Bo; Li, and Tak-Shing Peter Yum. Coolstreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies., 3:2102–2111. II-B

o Verifica-	SHA-1 hash; public key signing for live streaming	Unknown	Unknown	Unknown	Base (implicitly through decryption), enhancement (unknown)	Unknown
- Video						Unk
Video Transmis- sion	TCP, Pull; Unicast	TCP-Friendly Rate Control (TFRC); Pull; Multicast	Mostly TCP; Pull; Unicast	Almost exclusively UDP; Pull; Unicast	Base (Multicast), enhancement (Unicast)	Unknown
Video Schedul- ing	Rarest-first; helpers support collectors; Tit- for-Tat; hidden superpeers	Sliding Window, Least available first, buffer map (BM)	Sliding Window, buffer map, unclear chunk prioritisation, no Tir-for-Tat or derivatives; locally close peers with high bandwidth are prioritised	Unclear chunk prioritisation; location is not considered; high delay delivery	Base (cluster multicast), enhancement (synchronisation with base layer through buffering)	Buffer map
Content Discovery	Taste buddies	Unknown	Channels are discovered through central server	Channels are discovered through central server	Presumably through central server	Presumably through central server
Peer Organisa- tion	Tracker; mesh; taste buddies	mCache (unstructured)	Mesh; Tracker	Mesh; Tracker	Base (tree-based; special trusted layers; subclusters), enhancement (mesh)	Directed Acyclic Graph; locality first (low playback delay), bandwidth
Peer Communi- cation	BuddyCast; Tracker	Scalable Gossip Membership Pro- tocol (SCAM)	Gossip; channel tracker	Probably Gossip; channel tracker	Base (through cluster leader); enhancement (gossip)	Heartbeats (parent suggestion); two- hop neighbour list
Peer Identity	PermID (public key), exchanged via email	Unique node identifier (presumably the network address)	Unknown	Unknown	Base (via label and keys), enhancement (unknown)	Unknown
Join Procedure	Pre-known superpeers; open membership	Contact origin node of the stream; redirected to deputy; open membership	Central server gives channel and tracker list	Central server gives channel and tracker list	Base (CA gives time-limited label; session keys; cluster keys), enhancement (central Server)	Registration through RandPeer
Design Goal	Exploit social phenomena to increase usability and performance	Easy to implement, efficient, robust and resilient	Presumably scalability	Presumably scal- ability with low delay	Unprecedented security, scalability and heterogeneity	Provable network connectivity, failure resilience, efficient network resource usage
System	Tribler	DONet	PPLive	SopCast	TrustStream	DagStream

TABLE I P2P Streaming System Summary

			Main plane of attack		Attack Strategy	Position in Network	Attack Behaviour
Attack	Attack Summary	Defence Summary	Control or data plane	Target Systems	Independent or Collusion	Constant or Exponential	Constant, Increasing or Periodical
Sybil	Forge multiple identities	Trusted certificates, trusted devices, resource testing, recurring costs, observation	Data layer (achieve higher download speed by forging identities) or control layer(facilitate another attack)	DONet, PPLive, SopCast, DagStream	Only an independent attacker is needed, colluding attackers can verify their identities to honest nodes	Not important	Dependant on the goal of the attack
Whitewashing	Change identity after flagged as malicious	Similar to defences for the Sybil attack	Control layer		Independent though, colluding attackers can verify their identities to honest nodes.	Not important	Periodical
Flooding	Convince many honest peers to contact the target, thus reducing reliability	The target has to be able to discern flooding attacks	Control layer	Tribler, DONet, PPLive, SopCast, TrustStream (enhancement layer), DagStream	Independent for low resource targets, collusion for high resource targets	Not important	Usually target is only attacked once
Membership	Send forged routing tables, convincing honest nodes to form partnerships with malicious ones	Minimise influence malicious nodes on membership organisation	Control plane	Tribler, DONet, PPLive, SopCast, TrustStream (enhancement layer), DagStream	Collusion	Few malicious nodes placed choke points (connecting different parts of the network), collusion makes the attack more effective	Increasing
Omission	Drop complete packets or parts instead of forwarding	Make communication patterns harder to exploit (e.g. SecureStream)	Control plane, data plane (in multicast systems)	Tribler, DONet, PPLive, TrustStream (enhancement layer), SopCast, DagStream	Collusion	Few malicious nodes placed choke points (connecting different parts of the network), collusion makes the attack more effective	Dependant on the goal. Attackers can periodically drop packets, select few or all of them (black hole attack).
Pollution	Declare false content as available but transmit soiled content	Blacklisting, traffic encryption, hash verification, chunk signing	Data plane	DONet, PPLive, SopCast, DagStream	Independent highly effective, collusion even more so	Not important	Dependant on the intended degradation of the quality of service.

TABLE II
ATTACKS ON P2P STREAMING SYSTEMS SUMMARY

Development on Friend-to-Friend networks

Jan-Michael Heller

Abstract—This paper gives an overview over current development on Friend-to-Friend networks.

Common Peer-to-Peer applications either do not care about participants' anonymity at all or they implement techniques that guarantee anonymity against certain attack schemata. But they are still much more vulnerable to different attacks if the network contains a high amount of fraudulent nodes or if many nodes can be monitored or even manipulated by an attacker. Also membership concealment requires much more design work in those kind of networks.

Repressive associations that are powerful enough could hence manipulate the distribution of information or reveal the identity of many participants in such Peer-to-Peer networks.

A Friend-to-Friend network could solve this issues, because in such a network a node only maintains connections to trusted nodes. This allows to highly increase the possibility to repudiate the operation of ones node and to hinder a powerful attacker to manipulate or observate the exchange of information.

In this paper requirements on such networks are elaborated (eg. functionality, membership concealment, censorship-resistance) and existing approaches are analysed and compared, on how good they meet the requirements.

I. INTRODUCTION

PEER-TO-PEER (P2P) SYSTEMS have come to a wide usage the last decade, yet their operation purposes and technical possibilities are still not fully utilised.

Common P2P system for filesharing or multimedia streaming like, eg., BitTorrent, Skype, or SopCast, do not care about the anonymity of sender nor receiver, they are mostly built to deliver the best (respectively fast) service and be most fault-tolerant.

Mix networks go a step further: their purpose is to hide the senders or receivers identity. Such networks exploit the fact that they contain many - hopefully trustworthy - nodes. Therefore messages can be sent trough many different hops, which only have information about their neighbours and not the endpoints. When the message is also encrypted layer-wise, the path is called *onion route*. That way the identity of the communicating parties as well as the content of the messages is camouflaged.

Those networks still have the problem, that communication goes through many nodes of unknown trustworthiness and as soon as enough of them are fraudulent the identity of sender or receiver might be revealed.

An ideal Friend-to-Friend network (F2F network, often also darknet because of the ability of clandestine operation) tries to conquer this issue by only allowing direct connections to fully trusted nodes. This throws up new problems regarding the routing, as a node cannot simply connect to arbitrary nodes because it should know not much more about the topology than who its trusted neighbours are.

However, the *small-world-assumption* [1] encourages, that efficient routing can be possible in such kind of a network

because it is connected and every other participant is reachable in a (see also [2]).

So the routing algorithm is a central criterion when evaluating the functionality and scalability of such a network.

II. TERMINOLOGY

This section tries to settle some common terms in reference to F2F networks.

Most terms, such as anonymity, unlinkability, and pseudonymity are based on the definitions in [3].

Our environment consists of *sender*, *receiver* and *message*, to which nearly any communication-scheme can be generalised. An attacker cannot get information about sender nor receiver out of the content of the message. [3]

Most commonly, this is achieved by hop-to-hop anonymisation so that on a path one node always only knows the predecessor and successor for a certain message.

If the content of the message is sensitive the use of end-toend cryptography is also mandatory. For example, if Alice and Bob send each other messages through the network, and both have exchanged a key for usage of a secure cryptographic technique through a secure channel, they now can communicate encrypting their messages with that technique, so that no forwarder can get any information from that message. This is implemented, for example, in the private data-sharing of OneSwarm (see IV-C), in onion routing, or the ECRS algorithm of GNUnet (see IV-D1b).

A. Identity

Identity in common use (civil identity) is a linking between an existing person and a (hopefully) unique tuple consisting of its name, day of birth and city of birth.

Transferred to our scheme, the identity is the linking between the pseudonym of the node (the identifier) and a tuple of properties assigned to the machine it is running on. Such could be:

- IP-Address
- owner (and his civil identity)
- location
- operating system, hardware-fingerprint etc.

B. Pseudonymity

Identifiers which do not directly reveal the identity of a given subject are called *pseudonyms*.

Transferred to F2F networks a pseudonym is an identifier to address a sender- or receiver-node in the network.

C. Anonymity

Anonymity in our scheme means that it is not possible for an attacker to get knowledge about the identity of sender nor receiver from a given message or of a certain amount of captured data as being a transfer node in communication. Hence, it is important for a F2F network to keep its structure secret because the more the attacker know about it, the higher the possibility is for them to guess the identity of participants (that means breaking the membership concealment, see III-D).

This also highly depends on the operation of the network in detail, but in F2F networks in general, anonymity is mostly covered by the assumption, that the trusted neighbours are not fraudulent.

A much more detailed grading of anonymity is found in [4], which is not covered in this work.

D. Unlinkability

"Unlinkability of two or more items (e.g., subjects, messages, events, actions,) means that within this system, these items are no more and no less related than they are related concerning the a-priori knowledge." [3]

In F2F networks this chiefly means, that the friend relation between two nodes is not detectable by an attacker.

Unlinkability can also mean not to know, who is the sender or the receiver of a message.

E. Definition F2F network

A network consists of a set of nodes \mathcal{N} and a set of data items \mathcal{D} . Every node $i \in \mathcal{N}$ stores some data items $\mathcal{D}_i \subseteq \mathcal{D}$. A node which is in a friend relation with i is in the *friend subset* F_i of i. The friend relation is commutative, ie. $j \in F_i \Leftrightarrow i \in F_j$, but it is not transitive. [5]

Nodes only exchange information with nodes in their friend subset F_i .

A central proposition to fulfil the security objectives is the trust assumption:

Trust assumption:

It is expected that the real person behind a node only sets its node into a friend relation to persons that are not likely to give away information they get from being part of the network and also forward every message that passes correctly with regard to routing and without modification.

It is also assumed that all nodes are of integrity, meaning that they are not compromised with any software giving away internal data of the node.

With that assumption, it is not possible for an attacker to harm the network, if they are not able to modify or drop messages on the way between two nodes.

As other P2P networks F2F networks can have different purposes like data sharing, data storage, connection tunnelling, or media streaming, which all can be generalised to the previously (II) stated sender, message, and receiver scheme. However tunnelling or streaming applications need other strategies of maintaining data and distributing data because this content is needed for a short amount of time only.

F. Attacker model

An attacker is someone, who wants to break one of the stated security objectives, like anonymity or unlinkability, thus needs to obtain knowledge about the identity of a node or the friend relation between nodes, which form the overall structure, or he just wants to bring down the network at any cost.

Security paradigms can be measured by the kind of attacker, that is needed to successfully break those.

- 1) Trivial attacker: The attacker only maintains a single or few nodes that have at least enough friend relation to operate normally in the network.
- 2) Advanced attacker: An advanced attacker can instantiate many nodes which are more or less integrated into the net, but in a way that each node maintained by the attacker can communicate with at least one normal node.
- 3) Powerful attacker: This attacker can additionally monitor and modify the network traffic of many nodes. Such could be employees of ISPs or governmental organisations.

III. PROPERTIES OF F2F NETWORKS

In this section we introduce some fundamental terms which will be used in section IV to analyse and compare existing F2F networks.

A. Functionality and Scalability

The routing-algorithm plays a central role in the functionality of a F2F network. The structure of a F2F network can hardly be influenced, as it is a result of the user's choice of trust relations.

To measure the scalability, it has to be determined in which complexity a path to a desired node is found.

B. Censorship-resistance

In an ideal network there is no possibility that an attacker can hinder other nodes in transmitting special messages or store/receive certain data.

A network is called censorship-resistant to a specific attacker scenario, if it is not possible to prevent some nodes from receiving certain data that would be available under normal circumstances.

Censorship-resistance does not mean that it is not possible to bring down the whole network to stop the spreading of certain data.

C. Unrevealability of structure

The unrevealability of the structure can be measured by the kind of attacker, which is needed to uncover the identity or the friend relation between nodes of parts of the network. It also matters to which kind of other vulnerabilities such information can lead.

D. Membership concealment

Due to the nature of F2F networks, it is possible to completely obfuscate the use of the network because, if communicating parties are trusted, they can react only to messages that consist of a pre-shared secret, so that non-trusted entities do not get an answer and thus even the advanced attacker cannot detect the operation of a node in such a network.

Assuming that Eve can monitor the traffic, she could still guess from a used protocol that a machine is part of the network. Therefore the traffic has to be fully encrypted, or encapsulated in more inconspicuous protocols (eg. HTTPS).

IV. EXISTING APPROACHES

Over the years, many people have been trying to implement very different kinds of P2P networks, which fulfil our security requirements more or less. Many of them did not start with the intention of being a real F2F network and there are many approaches which meet our requirements at different levels but are no real F2F networks, as they do not rely on trusted connections.

Many widespread anonymising networks use untrusted mixes (see [6]) which still have problems assuring our safety objectives because an advanced attacker could easily undermine the network with many nodes and thus reveal the structure of the network.

In this section we analyse different practical approaches which more likely meet or F2F definition:

First the projects are sketched, then the networks operating mode is described (*principle of operation*). After that, the *functionality* (III) is analysed and afterwards the results are gathered and critically reviewed (*conclusion*).

A. Turtle

Turtle is the first P2P network that focused solely on trust based connections. It allows connections only to directly trusted nodes and forwards messages to other nodes in a mixlike fashion. Turtle is aimed to receive files from remote nodes that match certain search criteria.

1) Principle of operation: Every node maintains a list of friend nodes, to which messages can be forwarded. The trusted nodes are connected via a cryptographically secure connection, which is established by a preshared secret.

If a node searches for a file, it generates a query out of the properties it is looking for (logical conjunction of *attribute=value* pairs). Then a query identifier is generated out of the hash of the query-string and a random number to assure that queries do not collide. The query also includes a TTL counter which counts the hops the message has taken.

Every node maintains a table in which queries are registered, so that later on responses can reach the node, which initiated the query.

Nodes which receive query check their local file database, if it contains files that match the query. If so, a full attribute set of the found data is sent back to the querying node via the links that the intermediary nodes keep. The response message also contains a counter, which counts the number of hops.

The querying node then chooses which files it wants to receive from remote nodes:

First it distinguishes different and equal responses and lets the user choose, which one he wants to obtain.

Then responses with the lowest hop counter are chosen as sources and the querying node then sends requests through the paths to the sources.

Data are then return on the same paths the query initially took. [5]

2) Functionality: In small scale networks Turtle might perform well. But as the query paths have a hop complexity $\in \mathcal{O}(n^C)$ (where n is the number of friendship connections and C the hop counter) finding desired files in a network with many different files and many participating nodes will get too expensive.

Also distinguishing files only by their attributes might be problematically because that way different answers might get mixed up. This depends on the implementation (the clients should always attach hashes of the files as attribute).

3) Conclusion: Turtle was a pioneer work to build a real F2F network and it is a good starting point, to understand which issues matter when constructing such a network.

Because of it scaling problems it is probably not made for real-world usage. Also the hop counters, which packages on path construction contain, might get attackers too much information about the structure of the network (this was already known to the authors).

B. The Dark Freenet

As other historically grown networks, Freenet did not start as a distinct F2F network, it was just built to be "a distributed information storage and retrieval system designed to address [...] concerns of privacy and availability" [7]. Freenet is the one of the most researched on F2F network, thus it still also allows untrusted connections: Nodes can choose if they allow trusted connections, open connections or both.

1) Principle of operation: Freenet is built similar to Dots. Every file stored in the network has a key, by which it can be found and which is also used to proof the authenticity of the stored file. Keys are basically found by a routed depth-first-search with backtracking [8]:

Each node gets a randomly chosen location to which the keys of documents are associated. When storing (put) or requesting (get) data, the routing algorithm always tries to find the location closest to the documents identifier. It therefore chooses the neighbours with the closest location and delegates the request to it.

The routing then is done by consecutively delegating the request to the following neighbour, avoiding recently processed nodes, until the request is fulfilled or a maximum number of hops is reached. If the request can be met, the data is returned on the same path. Data are additionally cached at all nodes which are passed by it. [9]

That way a high redundancy of data is achieved, which helps to solve the problem of node churn and also spreads data in the near of the location where it should be originally stored, so it is found by less hops. If a get message gets stuck, meaning no nearer location is in the node's neighbourhood, a backtracking is performed.

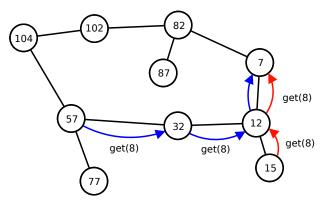


Fig. 1. get in Freenet

The document identifier is a so called "Content Hash Key" (CHK) which is a hash value generated out of the document. A part of the CHK is then used to encrypt the document itself, making it impossible to find or analyse a file without knowing its CHK. [8] That plays a big role in Freenets censorshipresistance, yet the distribution of the CHKs is the bottleneck.

2) Functionality: In order to get the greedy routing described previously to work efficiently, the networks location identifiers have to be restructured. Nodes join the network with random location identifiers. [8] Ideally, nodes only connect to trusted neighbours, but the latter also have random identifiers. So if selecting the following neighbour when querying for a key is adequate, but due to the fact that identifiers in a neighbourhood are not close to each other, the distance to the searched key does not necessarily decrease with every step.

The workaround used for that problem in Freenet is that a node looks for other nodes once in a while, that it could swap the location identifiers with. The two nodes exchange the location identifiers of their neighbourhoods to calculate, if exchanging the identifiers would decrease their distances to their neighbours. If so, they swap their location identifiers. [9] At this point the cache of the two nodes becomes invalid, so the swapping has to happen in a regulated frequency.

3) Conclusion: Freenet is a distributed data storage system which is able to run with a high number of nodes and store many documents safely because its caching mechanisms can deal with a high churn rate (see also simulation results [8], with the in IV-B2 described modifications, the number of hops in the routing-algorithm converges to $\mathcal{O}(\log(n))$, which makes routing quite feasible even in large scale networks.

However, this point has great drawbacks on our security objectives, especially the unlinkability of nodes.

A trivial attacker could easily request a node in its scope to swap the location identifiers with it. This node would send them all identifiers of its neighbourhood.

Also this problem can be used by an even weaker attacker, to bring down the whole network, by pretending to have a good situation for a swap, giving them false location identifiers (eg. the same to all nodes).

Pseudonyms in Freenet are the location identifier, discussed above, and a more static, unique node identifier, which does

not play a big role in the routing algorithm.

Assuming Freenet was an exclusive darknet (all nodes which do not allow connections from the opennet form such one), only direct trusted neighbours would know your identity (in this case IP-Address) and unless one of them is untrustworthy, nobody would get to know your identity except them.

As data are always cached when they trespass a node and paths are unknown to an attacker, at least an advanced attacker is unable to hinder the spreading of certain documents.

C. OneSwarm

An relatively new aim at data-sharing is OneSwarm. It is a combination of a Friend-to-Friend network, used for sharing data with restricted access combined with a mix-"like"-network for anonymous and public data-sharing. It accessorily provides an interface to the popular P2P file-sharing network BitTorrent.

OneSwarm is designed very purposeful, implementing techniques which make it easy for the every-day-user to become a part of the network. Such are automatic key exchange via email, XMPP, and local network or central maintained trusted and untrusted groups (see [10])

Restricted files are shared more likely in a direct connection, only identifiable and decryptable, if offered by directly trusted neighbours in the F2F network, whereas public files are distributed unencrypted through paths in the mix-network (preferably with most hops inside the F2F mesh).

OneSwarm makes use of a routing algorithm which needs to be discussed in-depth, especially for its scalability as it does not use any kind of identifier for nodes, rather it routes "at random" (see IV-C2a) and by capacity of the neighbours.

1) Principle of operation:

a) F2F network: OneSwarm builds its trusted part of the overlay by maintaining a DHT in which every node stores its identifier linked with a list in which its IP-Address and TCP-Port (identity) is encrypted with each public key of other friend's nodes. The encrypted identities are indexed by pairwise pre-shared secrets. [10]

The F2F network is used for the sharing of data with restricted access, and links of it are preferred for the routing inside the mix-network described below in IV-C1b.

Files with restricted access are encrypted symmetrically with a unique key [10]. If two trusted peers connect, they send each other a list of files they are allowing each other to access. This list contains the key needed to decrypt those file, which is encrypted with the public key of the receiver.

b) BitTorrent-like mix-network: Untrusted peers find each other via central community servers, which store the public keys as well as the IP-Addresses (identities) of those. [10]

Data is identified by hashes and sources for those are found via a special "congestion aware" flooding-algorithm:

A file is hashed, and parts of the hash and a random searchidentifier are combined to a query and sent into the network. The flooding algorithm uses the "shortest" path available by avoiding busy nodes to forward the query. It preferres trusted nodes, but also probabilistically chooses untrusted nodes to forward the requests to. [10]

Instead of using a time-to-live counter, like used in many other flooding-algorithms, the nodes maintain bloom filters in which the search identifiers they already saw are registered (see IV-C2a). This ensures that the path to the requested data does not contain any loops. However, this practice does generate a lot of useless connections, as a node forwarding a query can not exclude nodes which already saw the query before.

When a source of a file is found, it gets obscured by establishing a classical, unencrypted mix through the path explored by the request routing. Each peer only saves the identifier of the search request and the peers, which it got the message from and to which it sent it. Additionally a reply contains a path identifier, so that for example a node which forwarded the original request to multiple other nodes can later distinguish between the path where further communication is sent through.

To hide the fact that a peer holds the data itself from untrusted nodes, it randomly delays the response for a request. [10]

2) Functionality: It has yet to be shown, that OneSwarm scales to a large real world network where many files can be distributed simultaneously. In the previously cited paper, some test are described (see [10]); one which compares the transfer speed of files in a via Tor tunnelled BitTorrent Network and to Freenet (which is not originally meant to store large files, rather than information) and another that simulates a network where many small files are requested at different times.

Unfortunately the tests are not precise described and not enough data are given to draw clear conclusions from them.

The problem of flooding algorithms is definitely known to the authors of the paper:

"Although the majority of data transferred is due to popular objects, the majority of control traffic stems from requests for unpopular object for which search messages are forwarded to a large number of nodes in the overlay (during periods of low contention). This is an explicit design choice to improve availability without compromising privacy [...]" [10].

Even if the flooding-algorithm works feasibly in the test (under the assumption that the tests are significant), there still will be a big problem with this in real world usage:

Because that the tests are made on a basis that more than a third of the overall nodes is connected to at least a tenth of peers of the whole network, there will be problems in scaling that structure to a much bigger level because the peers will not be able to handle exponential complexity to build paths in bigger networks.

a) Practical analysis: Results from reading the source-code as of the commit¹ from January 4, 2012:

In SearchManager.java a set of 4 bloom filters is used. A passing search is continually added to the youngest of those. After every 5 minutes², the eldest filter gets deleted and a new one is created.

¹github.com:CSEMike/OneSwarm commi 548a27d05c35939417a7fe8503e61bbc2cbe5284

²see oneswarm_F2F/src/edu/washington/cs/oneswarm/F2F/network/ SearchManager.java:97, 170, 1600, 1657 for the rotation function and the decision, when to rotate and oneswarm_F2F/src/edu/washington/cs/oneswarm/F2F/network/ Overlay-Manager.java:745, 170 and 90 for the timer, which triggers the rotation Parallel to that, every search that is forwarded is inserted into a normal hash table³ and the method forwardSearch⁴ decides not to forward a message if it is on this table or (\lor) if any of the above created bloom filters gives a positive result for the id.

So the only thing the bloom filters do, is to create false positives, so that sometimes search messages are not forwarded (searches with equal IDs are filtered before according to the hash table). The bitstring filtered in the bloom filters consists of the search identifier and the hash of the searched files⁵. The bloom filter implementation itself is missing, so no analysis could be done, as to the frequency in which false positives do occur. (This "functionality" could however lead to a better scaling of the network, yet the better it is for the scaling, the worse seldom files would be found).

b) Security in F2F means: The grade of anonymity achieved in OneSwarm is similar to those in popular mix networks, except that there is no adequate end-to-end-encryption, if data is publicly shared, so that a node which is in the path between a sender and a receiver of a message always can see the content of that message, and thus potentially manipulate it, which increases the number of possible attacks.

When the attacker has the capabilities to instantiate many nodes in the network (*advanced attacker*), they then could possibly reveal paths in the mix-network part and also hinder the distribution of special files (censorship).

Another problem is, that if the forwarding of certain data is a liability, a node's possessor could be held responsible because his identity is revealed to any other node he connects to in the mix network and the data are only encrypted per hop, so if the next hop is the attacker, they could proof, that the node forwarded the traffic.

Moreover, the combination of mixes and trust relations could make it possible for an advanced attacker with many nodes, to reveal the friend relations of nodes because they are preferred when routing data. However this is obfuscated at least for a short-time attack because OneSwarm also tries to reuse connections to untrusted nodes.

The source of data can also be determined by a similar attack because if an attacker is connected with two nodes e_1 and e_2 to a node i, which both get forwarded messages from i, e_1 could send a request for a certain file and if e_1 gets a response without e_2 getting a forwarded query for the same file, she can be sure that the request is met by i.

If sufficiently trusted connections exist and a user chooses to not participate in the public mix network, it is possible, under the general trust assumption, that a high grade of anonymity is reached.

Censorship-resistance only in the F2F network is comparable to Freenet, but the fact, that information does not get cached at many different nodes on a path raises the ability of an attacker to stop the availability of certain seldom

³see oneswarm_F2F/src/edu/washington/cs/oneswarm/F2F/network/ SearchManager.java:373

⁴see oneswarm_F2F/src/edu/washington/cs/oneswarm/F2F/network/SearchManager.java:346

⁵see oneswarm_F2F/src/edu/washington/cs/oneswarm/F2F/network/ SearchManager.java:374

information by withdrawing single nodes from circulation. Also publicly shared files could be easily filtered out by intermediary nodes, as they are not encrypted.

3) Conclusion: The idea of combining mixes and F2F architecture brings many advances for new users, as they can participate easily in the network, without having many friend relations in the beginning.

However, there are still some major issues regarding the routing and the security of connections when combining hops in the F2F overlay and the mix overlay to one path. Also some encryption ideas provided by networks like *Tor* should be implemented.

Some more attacks have been found recently by a conference paper, which is not mentioned here because it does not meet the authors claims on neutrality and independence.

As the public sharing somehow still relies on the distribution of the BitTorrent file-definitions, censorship-resistance of this network is disputable.

D. GNUnet

GNUnet started with the idea of anonymous and censorshipresistant data storage and has come to a lot of changes resulting of related scientifical work in the last decade. Also it is known to be well documented which makes it easy to implement enhancements.

The idea of GNUnet is to provide an anonymous and censorship-resistant network which is able to provide various services (file distribution as the common use). Its focus is on privacy, while performance is only aimed to reach a good grade of functionality and scalability. [11]

It also allows to be used as a F2F network only, mixed with an opennet or completely without F2F connections.

- 1) Principle of operation: To identify nodes, every peer of the network has an asymmetric key pair. The hash of node's i's public key is used as an identifier of the node. That way, every peer j that has the public key of i can verify its identity by a simple challenge-and-response principle. If i and j want to communicate, each of them generates a symmetric key. Those are exchanged encrypted by the public key of the counterpart. [12] GNUnet implements novel techniques to achieve anonymity or censorship-resistance.
- a) Routing algorithm: GAP: This includes gap, a routing algorithm which works with source rewriting as in normal mix cascades but allows to evade the rewriting, if a node is too busy handling traffic, thus giving it the ability to redirect traffic around it. [13])

To assure anonymity different queries and replies are always combined to groups, so that the originator of a certain message is obfuscated. [13]

Queries are forwarded depending how much load is on the forwarding node. The more it is busy, the less other nodes it is forwarded to. To which nodes it is forwarded to is decided with help of the distance of the queried key to the key of node, the previous utilisation of the path (more used paths are favoured) and at least a random factor.

Messages are put on a buffer and forwarded together after a random amount of time, or when the buffer is full, to complicate timing attacks. Gap does not implement any techniques to avoid loops in forwarding and does not guarantee the delivery of messages.

b) Encoding for Censorship-Resistant Sharing (ECRS): GNUnet makes use of a special encryption scheme called Encoding for Censorship-Resistant Sharing (ECRS), which is explicit described in [14].

ECRS implements a keyword search which allows users to use natural language for searching, superseding document key distribution like in Freenet. It also tries to make data confidential, but at the same time allowing each intermediary node to check the integrity of the parts of the document using the query. The encryption is done deterministically, so that every block of an equal file is translated to an equal ciphertext as well to support swarming.

The main difference to Freenet is, that not the whole document is hashed and ciphered but split into blocks for which the CHK is generated. Those blocks are then encrypted each for itself and are all verifiable by their CHK by oneself.

A public and private keypair is deterministically generated out of the hash of the search keyword (The hash of the keyword is used as randomness input to the keygenerator). The hash of the generated public key is then used as a hash for the query. The metadata that are searched for are encrypted symmetrically by the hash of the keyword. This ciphertext is signed using the public key generated before. (see Figure 2) The whole query consists of the query hash, the encrypted metadata and the signature for them. [14]

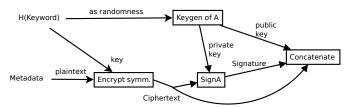


Fig. 2. Keyword search with ECRS [14, Figure 2]

That way, every intermediate node can validate the content of the query against the public key, but only those, who know the keyword, can decrypt the whole query.

A censor could thus only block traffic for which every possible keyword is known.

2) Functionality: GNUnet did not initially aim to be a F2F network and there do not exist deeper analyses of the F2F only use of this network.

GAP might work well to speed up the connections in a more public network with sparse F2F connections, but in a network which tends to rely solely on F2F connections, routing around Peers might get problematically, because it requires knowledge about the neighbourhood of the peer which should be avoided.

Lately a routing algorithm called R^5N has been proposed by Evans and Grothoff (see [15]) which is built to work on small-world networks. This algorithm could improve GNUnet's F2F capabilities if implemented in the future.

On the other hand ECRS is ready to be used in such an environment. It is a good way to communicate confidentially with other nodes, which know about a certain secret keyword.

3) Conclusion: GNUnets ECRS is a very powerful mechanism to combat censorship. If the network is used as a F2F network it becomes extremely hard to block specific content or to even guess which content is transferred.

The routing algorithm may still has some weaknesses regarding anonymity which could also be avoided, if friendship connections are preferred; even though common timing attacks are not possible.

V. SUMMARY AND CONCLUSION

F2F networks could provide a strong enhancement in the realisation of the safety objectives in Peer-to-Peer networks, yet trade-offs always have to be accepted.

The Combination of darknets and opennets strongly supports new users participating in such networks and is thus needed to bring a network to success. However, as seen in OneSwarm, such combinations must be handled with care because they could reveal too much information to fraudulent peers.

The problem of efficient routing whilst not revealing too much sensitive information has not been resolved perfectly yet. Freenet offers good techniques in routing, but the lack of structure in a F2F network is thwarted by reorganising the identifier which in its current implementation reveals too much information about the nodes friendship relations. GNUnet offers some interest techniques, which enhance the censorship-resistance, but it still has to be shown that its routing algorithms would efficiently work in a large scale F2F only environment.

As the example of OneSwarm has shown, optimising such a network in performance is possible, but this can have great drawbacks in privacy if it is done too friviously. Cryptography provides potent assumptions which, if applied properly, can have great impacts on privacy.

The observation of different systems has shown, that the best approach is to assure that security constraints are still met, while thinking about other enhancements.

GNUnet is assumed to be the most seminal F2F network because it already has strong - yet not scalability defacing - policies assuring anonymity and censorship-resistance, even without the trust assumption of F2F networks and is constructed most flexibly allowing the fast deploying of different new services.

F2F networks could give the people a great opportunity allowing to bias the distribution of certain information on a basis of public agreement, avoiding single powerful censors who control the many.

REFERENCES

- [1] D. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, 1998. I
- [2] O. Sandberg, Distributed routing in small-world networks, 2005. I
- [3] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity - a proposal for terminology," in *Designing privacy* enhancing technologies. Springer, 2001. II, II-D
- [4] C. Diaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," 2002. [Online]. Available: http://www.cosic.esat.kuleuven. be/publications/article-89.pdf II-C
- [5] B. Popescu, B. Crispo, and A. Tanenbaum, "Safe and private data sharing with turtle: Friends team-up and beat the system," in *Security Protocols*. Springer, 2004. II-E, IV-A1

- [6] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, 1981. IV
- [7] I. Clarke et al., "A distributed decentralised information storage and retrieval system," Ph.D. dissertation, Master's thesis, University of Edinburgh, 1999. IV-B
- [8] I. Clarke, O. Sandberg, M. Toseland, and V. Verendel, "Private communication through a network of trusted connections: The dark freenet," 2010. IV-B1, IV-B1, IV-B2, IV-B3
- [9] N. Evans, C. GauthierDickey, and C. Grothoff, "Routing in the dark: Pitch black," Ph.D. dissertation, University of Denver, 2007. IV-B1, IV-B2
- [10] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson, "Privacy-preserving p2p data sharing with oneswarm," ACM SIGCOMM Computer Communication Review, 2010. IV-C, IV-C1a, IV-C1b, IV-C2
- [11] Anonymous. last accessed 10. January 2012. [Online]. Available: https://gnunet.org/goals IV-D
- [12] —. last accessed 10 January 2012. [Online]. Available: https://gnunet.org/concepts IV-D1
- [13] K. Bennett and C. Grothoff, "gap practical anonymous networking," in Designing Privacy Enhancing Technologies, Springer-Verlag. Springer-Verlag, 2003, p. 141–160. [Online]. Available: http://grothoff.org/ christian/aff.pdf IV-D1a
- [14] C. Grothoff, K. Grothoff, T. Horozov, and J. Lindgren, "An encoding for censorship-resistant sharing," Tech. Rep., 2003. IV-D1b, 2
- [15] N. S. Evans and C. Grothoff, "R5n: Randomized recursive routing for restricted-route networks," in *Network and System Security (NSS)*, 2011 5th International Conference on. IEEE, 2011. IV-D2

A Survey: Monitoring and Measurement Techniques in Peer-to-Peer File-sharing Systems

Gerold Hintz

Abstract—Analyzing Peer-to-Peer (P2P) file-sharing systems is an important task to improve their performance and is essential to their development. Due to their distributed nature, empirically gathered measurements are the only way to make accurate statements about state and behaviour of a P2P system. However, depending on the restrictions of a given network, gathering this data is not a trivial task. This paper studies and categorized several monitoring techniques that have been applied in the past. As these techniques are vastly dependant on the architecture of the P2P system, this paper will focus on two different P2P systems being used for file-sharing in order to cover a wide range measurement techniques. The Kad network will be introduced as an exemplary structured network, while the BitTorrent protocol will serve as an example for unstructured networks used for filesharing. This paper will define characteristics and key properties of P2P systems, and examine the key challenges in measuring them. Metrics to assesses the performance and efficiency of measurement techniques are introduced, and problems of applying these metrics are discussed. Finally, the paper will give a conclusion on the current state of measurement techniques.

Index Terms—Measurement techniques, structured, unstructured, Kademlia, BitTorrent, Crawling, Monitoring

I. Introduction

P2P systems are an architecture to connect equally privileged peers with each other, to distribute their resources to the whole network¹.

File-sharing is one of the most widely applied usages of such networks: A set of peers connect the network to share files without the need for a centralized server. The advantage of such an architecture is that clients provide bandwidth, storage space and computing power, which makes the distribution of content cost efficient. If such a system also assures redundancy of data across nodes, it also becomes more reliable, as there is no single point of failure. Although there are a couple of other uses of P2P systems - such as multimedia streaming² distribution of data is by far the most common use case of P2P to day. Measurements³ are an essential aspect to the analysis and development of P2P systems. This paper will motivate the need for such measurements, defines key properties which can be measured, and introduces metrics to assess the performance of measurement systems. The measurement techniques being employed are vastly dependant on the P2P system in question. In order to cover a wide range of such techniques, this paper

¹While P2P *network* refers to a specific instance of such interconnected peers, the term P2P *system* not only incorporates the network, but also the client software involved, and the underlying protocols.

introduces two of these systems as examples. *BitTorrent*⁴ as an unstructured system, providing a centralized service for peer handling, and *Kademlia*, a structured, truly decentralized P2P system which establishes an overlay between peers. For the purpose of this paper these two systems are considered to be representative for the majority of P2P file-sharing systems in existence. The remainder of this paper is structured as follows.

Section II will start by explaining the background of P2P systems. The difference between structured and unstructured systems is explained. For structured systems, the widely employed concept of a Distributed Hash Table is explained and Kademlia and BitTorrent are briefly introduced as P2P architectures for the above-mentioned reasons. Section III will then motivate measurements and outline characteristics of the performance of P2P systems. In Section IV the properties which can be measured are defined. In Section V the different techniques of monitoring are introduced and their applicability as well as advantages and disadvantages will be discussed. Section VI subsequently introduces different approaches to aggregate the data being monitored. Evaluation and metrics for assessing the performance of measurement techniques are discussed in Section VII. Finally, the observations of this paper are concluded in VIII.

II. BACKGROUND

P2P systems can foremost be divided in *structured* and *unstructured* overlays (interconnection between nodes).

- a) Unstructured systems: Peers are connected arbitrarily. In such a network, a node can not address another determinate node in the network; resource location can either be done by flooding a query through the network or it relies on some external mechanism; BitTorrent simply creates a different overlay for each resource (described in Section II-C).
- b) Structured systems: A well defined overlay is established between peers and an algorithm is used to effectively route traffic. An addressing scheme is used to address every single node in the network and a consistent protocol ensures that data can be stored and located anywhere in the network. The most common architecture of structured systems are implementations of the DHT interface.

A. Distributed Hash Table (DHT)

More than a decade ago, the DHT was introduced as an abstract concept for storing and retrieving content among decentralized peers in a structured network. A (key, value)

⁴Here, *BitTorrent* refers to the unmodified version of the protocol, which seperates the concern of resource location to an external mechanizm.

²"Skype" is the most noteworthy exception of a P2P system, which streams audio and video between peers. However, due to its proprietary client and protocol, it has never been studied in-depth.

³The terms *measurement* and *monitoring* will be used interchangably in this paper.

tuple can be stored in a network of nodes, which are connected via an overlay. The key is obtained by applying a hash-function to the value. Each node participating in the network can then store or retrieve a tuple into the DHT. At the abstraction layer of a DHT, no assumption about the layout of the overlay or the routing therein is made. A number of implementations of DHTs are used in existing P2P systems, most prevalently Chord [1], Pastry [4], and Kademlia. The latter will briefly be described in more detail as an example structured P2P system.

B. Kademlia

Kademlia [15] is a protocol implementing a DHT. Each node has an identifier called the KAD ID, which is an element from the key-space. The width of the identifiers is set to 160 bits and is randomly generated for new peers. It is persistent between multiple sessions, meaning a peer can leave and rejoin the network and re-assume its former position in the network. Routing in Kademlia is based on a distance measurement between two peers a and b known as the XORmetric: $d(a,b) = a \oplus b$ which simply takes the bitwise exclusive-or of two KAD IDs and interprets the resulting value as an integer. This metric is 0 when a = b, it is symmetric and it respects the triangle inequality. A peer uses this metric to store its *contacts*: For each bit index (0..159) up to k contacts can be stored. The parameter k is known as the *bucket size* and is set to k = 20. Using these *buckets*, an unbalanced routing tree is built and a peer will remember many contacts close to their own KAD ID in the key-space, and fewer contacts that are far away.

Storing and retrieving objects in the network then works by hashing the data (either the file-name or its content) and then finding the node whose $KAD\ ID$ is closest to the hash. The protocol messages are straight-forward. A Find message is used to locate nodes close to a given key. A recipient will answer with the k closest nodes it knows. This process is repeated with the new contacts until the closest node is found. After location, it will be followed by a Get message to retrieve a value from a target node, or a Store message to publish a new value to the network. To add redundancy to the network, a given (key, value) tuple is stored not only at the closest node, but also a parameterized number of nodes around this key.

An important property of Kademlia is that any information about new peers that is gained through lookup or passive routing is added to the contacts of a peer, if there is still sufficient space in the corresponding bucket. Lastly, a *Ping* message can be used to verify the aliveness of a contact, which is done periodically. Contacts that do not respond are removed.

A notable implementation of Kademlia is the *Kad network*, a famous, albeit controversial file-sharing network. A number of clients acting for this network exist, an early adaptation was Overnet [14], and most famously the open source client eMule [2] and its derivates. It is also worth noting that the malicious botnet "StormWorm" [19] uses Kademlia for decentralized communication.

C. BitTorrent

BitTorrent is arguably the most popular P2P file-sharing system in existence, and amounts to 40% to 70% of all internet traffic [22]. The protocol itself provides no means of searching or locating a resource. For this purpose, a variety of web servers have been developed, which provide a centralized server-client architecture for searching and downloading metainformation in form of a .torrent file. Using this file, a peer can then contact another centralized service called the tracker, which is responsible for handling the arrival and departure of every peer. The tracker can supply a client with a list of other peers, effectively creating the overlay structure between the nodes. Using this server-P2P hybrid system, each file (or fileset) has its own network with unique nodes (although one tracker can be responsible for multiple *torrents*). While the tracker is aware of all nodes in the network, as well as meta-information about their download progress, each peer only knows the subset of nodes that the tracker has revealed to it. Communication with the tracker is based on the tracker protocol which uses HTTP GET messages to query the tracker for information. Peers communicate with each other by using the peer wire protocol which lies directly on top of TCP: A single file is split into multiple chunks, which are exchanged between peers one-by-one. The messages of the peer wire protocol only have to take care of of exchanging the information which chunks of the file are required. This is done by sending interested and not interested messages with a bitfield. To control downloading, BitTorrent uses a system called *choking*. A choked peer is not allowed to download, and only a limited number of peers are unchoked at a time. For this process, choke and unchoke messages are used, only unchoking peers that offer data.

To ensure operation of the network, sharing in *BitTorrent* is done in a tit-for-tat fashion: In general, for each chunk that has been received by a peer, another chunk will be offered to that peer. In order to bootstrap new peers into the system, *optimistic unchoking* randomly also unchokes peers that do not offer any data. For the actual transfer of chunks, a *request* message is issued which will cause the transfer of the payload. After the *SHA-1* hash of a completed chuck is checked by the downloading peer, it sends a *have* message, indicating that the download has completed successfully.

There are some additions to the plain protocol in existence. For instance the *Vuze*⁵ BitTorrent Client [5] added the ability for each peer to act as a tracker, effectively turning *BitTorrent* into a decentralized network implementing a DHT. In the same way, Khashmir [3] extends the *BitTorrent* Protocol to create a Kademlia overlay. Peers that support Khashmir can seamlessly integrate with ordinary peers not capable of the protocol extension.

Another modification some networks employ (transparently) is an enforcement of a certain upload-ratio. Each peer has to be signed in with an account, and the tracker verifies a sufficient share-ratio, or otherwise refuses its service to the client.

⁵formerly known as Azureus

III. PURPOSE OF MEASUREMENTS

A. Motivating measurements

As opposed to other services on the Internet, P2P networks often do not have an owner in the common sense, or any party being responsible for their operation. However, there are several reasons for monitoring these networks: Foremost, it is necessary for developers of P2P systems to analyze the behaviour of the network. As this is highly dependant on the behaviour of thousands of users, monitoring a real network is the only reliable source of information (with an unsatisfactory alternative being simulations). The objective is to generalize the findings to tweak the network's performance. Empirically gained knowledge is used to fine-tune a wide range of parameters of P2P systems. An example for such a parameter is the bucket-size parameter of KAD.

Understanding the dynamics of P2P traffic can also be a valuable information for network operators wishing to implement transparent caches. As user and system behaviours differ fundamentally from conventional web access, different caching strategies have to be adapted.

Another motivation for monitoring is related to the controversy of illegal files being prevailingly distributed via P2P systems. The owner of a copyrighted file might be interested in a wide range of data: Foremost, the amount and identities of unique peers sharing the file; but also statistical data like the geographical distribution of these peers. If, for instance, a large portion of illegal traffic can be geolocated to another country, legal prosecution can become unfeasible for a copyright holder.

A goal closely related to monitoring a network is also an intervention in its operation. A study by Thibault et al. [6] has partially been funded with the objective to locate illegal content and observe the related users behaviour - but also the removal of these files from the network.

Another problem arising in P2P networks is undesirable user behavior. Not all peers act for the benefit of the whole network, and some might violate protocol standards, effectively harming the whole network either for their own benefit or malicious intentions. Torres et al. [31] performed a study on detecting such malicious nodes through monitoring traffic.

A meta-purpose for many goals in measurements can be described as assessing the performance of the system.

B. Performance of P2P systems

To justify measurements of parameters in P2P systems, it is necessary to also define the characteristics by which a P2P system is evaluated. In particular, the demands of a P2P end-user and therefore the goals in the development of P2P systems.

a) download speed: The download speed of a single object in a file-sharing system is the most obvious metric of performance of a P2P network. Most P2P users want to minimize transfer times, sometimes with the secondary requirement to also minimize the number of simultaneous connections.

- b) availability: As resources in P2P networks can "die" (are no longer available, or in case of BitTorrent only partially available), the system must ensure redundancy of resources, which is evident as availability to a user.
- c) fairness: Fairness is a characteristic not directly important to a single user, but a requirement needed for a complete network to operate as desired. The general understanding is that a single peer should only consume as much as it is providing itself. In case of file-sharing, users should upload as much data as they download to ensure a balance and prevent "free-riding" (downloading without uploading).
- d) vulnerability: A P2P network should be resistant to malicious peers, disregarding protocol standards. The objectives of such peers could be free-riding, providing wrong or malicious data, hiding existing data from the network, or disrupting the operation of the network in general.
- *e)* search efficiency: As the lookup in P2P systems have a probability of failure/inaccuracy, the success ratio of lookups is an important metric.
- f) response time: As the time required to download a whole file in P2P systems is exceptionally long, often days, the response time here only refers to signaling traffic that is directly visible to the user. In a DHT this means the time delay between a lookup message, and the response about the availability of the resource. Obviously it is undesirable to have response times that hamper user interaction.
- g) resiliency: In networks in which peers rapidly join and depart, and effect called *churn*, the immunity of the network to dropping neighbours is described as *resiliency*. A key strategy to accommodate with this problem is for a peer to identify neighbours that are likely to be long-lived, and prioritize them. The observed result by a single user is a lower probability of peers becoming unavailable.

The list above includes both quantitative and qualitative characteristics. A qualitative characteristic, such as *vulnerability* can not be expressed in numbers, but has to be argued about. A quantitative characteristic can be expressed in numbers, either by directly observing it, such as the *download speed* of a certain transaction, or it can be inferred, such as the *average download speed* of all peers. Some qualitative characteristics can be *quantified* by introducing a metric; for instance the value *share ratio* = $\frac{\text{bytes uploaded}}{\text{bytes downloaded}}$ averaged over all peers, can be a decent quantification of *fairness*.

Everything that can be directly observed, inferred, or expressed through a metric is called a *property* in the scope of this paper. *Properties* will be described in the following section.

IV. PROPERTIES

There is a wide range of properties of a Peer to Peer system that can be measured. Due to different objectives, the majority of measurement studies have focused only on a small subset of these properties. Additionally, different studies disagree on the terminology. The same property might be referred by a different term, as well as the same term differing in its definition. This paper will make an attempt to give a comprehensive overview of all properties that have been monitored or mentioned in related works, and adapt the most

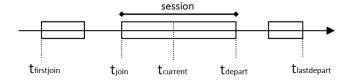


Fig. 1. Time spans of peer activity across multiple sessions

widely accepted definitions. Table I gives definitions of these properties and lists what studies have examined them. (Fig 1 acts as a reference to clarify the time frames of peer activity.)

Stutzbach et al. [28] break down these properties to be either static or dynamic: A static property can be observed at a specific moment t in time, while dynamic properties lack this temporal locality; e.g., the session length of a peer, which is defined as the difference between two points in time. Furthermore, all of the following properties can be observed for a specific entity e (e.g., a single peer, a single object, etc.). Global properties are a special case with the whole network being the only entity.

Most analysis of this data is interested in the mean (possibly also: median, min, max, sum) of these properties. This means that both in terms of time (average over all t) as well as locality (average over all e), a compound value is calculated from the measurements. The *mean* is therefore only included in the property name if there is an additional dimension of averaging a value.

While some of these values can be retrieved relatively easily, such as the size of a given resource, measuring some of these properties is very difficult. In the following, the key challenges of measurements will be examined.

A. Challenges

- a) Churn: Stutzbach et al. [29] have examined the collective effect called Churn, which describes the rapid arrival and departure of peers in a network. This is not only a problem for the operation of the network, but also for measuring it: Events, such as the arrival and departure of a single peer can be so fast that they are difficult to observe.
- b) Network failures: A peer which responds to a query is considered available. In case it does not respond it might not actually be unavailable, but due to common network failures is temporary unreachable. It is therefore important to set an accurate timeout for a peer to answer, as well as a possible number of attempts to contact it. Network connectivity failures can either be random, or systematic. In the latter case, a single peer might switch frequently from being available to being unavailable, a behaviour described as flapping [29].
- c) Edge effects: Since observations can only be made for a finite length of time, edge effects can distort the measurements. A monitor is oblivious of any property that spans more than the observed time window, e.g., long session lengths. This adds a bias toward short sessions. To cope with this problem, Saroiu et al. [20] have introduced the *create-based-method* in which the measurement window is divided into two halves, and only sessions that start in the first half are considered. Using

this method, an unbiased measurement of session lengths of up to half of the time frame is possible, while also identifying sessions exceeding this length. Although the mean of session lengths is still unknown, their median can be calculated with this method.

d) Peer distinction: One challenge in monitoring peer activity is to accurately detect unique peers. For instance in the case of Kademlia networks, the KAD ID is an intuitive characteristic to identify unique peers as it is intended to be persistent throughout sessions. However, it is not always reliable as Steiner et al. [26] have found a large fraction of peers changing their KAD ID for every session due to clients not following this specification. The IP address can only be used for peer distinction in some cases. Some Internet Service Providers commonly assign dynamic IP addresses which change on a daily basis. Another problem is that many peers are behind a NAT, resulting in multiple peers effectively sharing the same IP address. NATs and firewalls also create the problem of not being able to contact some peers directly. These peers communicate in the network only one-way by opening all connections themselves.

B. Further analysis

Once a set of data is collected, a measurement study then proceeds to do further analysis:

- a) Conditionalization: All of the previous properties, can also be analyzed under certain conditions or parameters such as time of day, day of week, or category of shared files.
- b) Inferred parameters: In addition to parameters which can be directly observed, values can be calculated from the dataset. For instance, by monitoring all signaling traffic between a set of peers, the (apriori) probability to respond to a message is an insightful value to examine, as done by Falkner et al. [8].
- c) Correlation: The listed properties can be analyzed for correlation to one another. Asserting one property as a function of another is done to make predictions based on data observed in real time, or infer on user and system behaviour. Possible correlations are:
 - uptime \mapsto remaining lifetime
 - The correlation of the observed uptime to the remaining lifetime of a peer is one of the most important theories to investigate. If a client can predict the remaining lifetime of a known peer it can make a more sophisticated choice of which neighbour entries to keep and which to discard.

traffic, and their demand declines as their lifetime grows.

- object size → percentage of bandwidth
 - To investigate traffic distribution in a network, it is useful to know what size of files cause what portion of traffic load. Assuming the percentage of global bandwidth consumption as a function of an objects size can give insight in what actually causes most workload. Findings suggest that the major load is caused by small files, which are often audio files.

TABLE I OVERVIEW OF P2P PROPERTIES

View	Property	Description	References
	degree	number of neighbours	[9] [7] [31]
	session length	[9] $t_{\rm depart}$ - $t_{\rm join}$. Some works [10] define this as a continuous timespan of activity	[8] [7] [10] [18] [27]
	uptime	$t_{ m current}$ - $t_{ m join}$	[<mark>9</mark>] [17]
	remaining uptime	$t_{ m depart}$ - $t_{ m current}$	
	lifetime	$t_{\rm lastdepart}$ - $t_{\rm firstjoin}$. (lifetime spans multiple sessions)	[27]
	availability	percentage of time being online	[10]
	activity fraction	fraction of time a peer is transferring content over its lifetime	[10]
Peer	portion of stale entries (neighbours)	The portion of neighbour entries of this peer which do no longer respond to queries.	[8]
	requests rate	average time between two <i>request</i> queries	[9] [8] [16] [10] [18]
	(total / up / down) transfer rate	transfer rate summed up through all neighbours	[9]
	mean transfer rate per connected peer	average transfer rate through all neighbours (mean of all one-to-one connec-	[9]
	mean transfer rate per connected peer	tions)	
	(total / up / down) bytes transferred	transfer rate summed up over time, for timespan Δt	[<mark>9</mark>]
	seed / leech ratio	bytes uploaded bytes downloaded. Essential for assessing fairness.	
	geographical location	geographical location of the peer, usually on country-level accuracy (determined by IP)	[18] [32] [27]
	concurrent transactions	number of concurrent transactions taking place (number of peers with which	[17] [10]
		payload is exchanged)	
	lingering time	(BT only) time since the peer has turned into a seed	
	response time	time until peer answers to a simple protocol message (ping) through the overlay	[9] [8] [7] [10]
		(or some other point of reference)	
	clustering coefficient	fraction of neighbours that are also connected to each other	
	size	file size (in case of BT also chunk size, etc.)	[16]
	availability	(BT only) percentage of the completeness of the file throughout all peers.	[18]
01: (01)	redundancy	number of complete replicates of the object throughout all peers (sometimes	[8]
Object (file)		called health for BT)	F101
	age	time since the object was first made available	[10]
	lifetime class	time until an object dies (is no longer available in the network)	[10]
	ciass	any type of classification on a shared object. e.g.	[17]
	manulanity.	classes like audio / video / executable, or flags like good / fake	[7] [10]
	popularity	index in a sorted ranking, e.g. by <i>number of global requests</i> download speed measured in bits / second	[7] [10]
	speed transfer time	the time until the complete payload of an object is transfered	[7] [10] [31]
Transaction	progress	progress of completion of the transaction (at time t)	[7] [10] [31] [17]
Tansaction	transaction size	if the transaction is terminated before completion, the actual size is the amount	[31]
	transaction size	of bytes transferred	[21]
	size/time quota	ratio of $\frac{\text{size}}{\text{time}}$ where size is the object size	
Global	distinct peers	the number of distinct nodes in the network (at time t , or timespan Δt)	[9] [8] [7] [16]
	distinct objects	the number of distinct objects throughout all nodes	[16]
	number of requests	number of request through timespan Δt	[9] [16] [10]
	number of transactions	number of transactions through timespan Δt	[<mark>7</mark>]
	proportion of successful transactions	percentage of transactions that have been completed	[<mark>9</mark>]
	bytes transferred	global sum of all traffic	
	content demanded	sum of the size of all objects requested	
	distribution of message types	distribution of signaling traffic in respect to the protocol. (e.g. percentage of	[9] [7] [32]
	peer arrival rate	lookup messages) average inter-arrival time between two distinct peers	[17] [10] [18] [27]
	object arrival rate	rate at which new objects are deployed to the system	[17] [16] [10]
	diameter	the longest shortest path between any two nodes of the overlay graph	[17] [10] [10]

• object age \mapsto popularity

Asserting popularity of an object to be a function of its age is done to understand the dynamics of popularity changes of files. Predicting the popularity development of an object is a valuable information for a network operator trying to add transparent caches.

d) Distribution functions: Properties are usually given as a "cumulative distribution function" from a captured dataset. It is typical to assert a distribution function to the resulting graph in order to generalize the findings to networks of arbitrary sizes. Session length has formerly be claimed to be heavy-tailed distributed [10], but newer studies suggest it to be

Weibull⁶ or log-normal⁷ distributed. [29].

V. MONITORING TECHNIQUES

Monitoring techniques can be split in two main categories: *passive* and *active*.

Passive techniques do not have an influence on the structure or operation of the network. This can be achieved either by intercepting traffic, or by analyzing logs. They have the obvious advantage of not distorting the operation of the network.

Active monitoring refers to actively probing the network. This refers to both querying a set of peers from one centralized

⁶http://en.wikipedia.org/wiki/Weibull_distribution

 $^{^{7}}$ log-normal is actually a special case of Weibull where k = 1

location, as well as distributed probing using one or more monitor nodes which are inserted into the system. Usually, some dedicated software is used for active monitoring techniques.

A. Passive monitoring techniques

- a) Traffic interception: If the monitoring party is an administrator of a network, they can use this to "eavesdrop" on P2P traffic. This is usually an ISP intercepting all outbound traffic at a gateway. The advantages of this system is a transparency toward the peers, and the interception of virtually all information passing in or out of the network. A downside, however, is the restriction to traffic. Idle peers that do not send any messages are invisible to this type of monitoring. Also the subset of peers behind a certain gateway might provide a biased image of the complete network. The notable studies [10] have been conducted at the University of Washington, whose network users are not representative regarding an average home-user. Another difficulty affecting the applicability of this method is the workload caused for traffic analysis. The first challenge is to correctly identify the traffic corresponding to the monitored P2P system, inspecting packets as least as possible. Depending on what data is being collected, deep packet inspection is still necessary for all P2P traffic, which causes great workload.
- b) Tracker log analysis: In general, P2P networks do not have any centralized authority to refer for global logs. However, in case of *BitTorrent*, a centralized server called the *tracker* is responsible for handling arrival and departure of all peers as well as their transfer progress. While this way only a small set of parameters can be measured, this monitoring technique has essentially no overhead.
- c) Tracker querying: If the monitoring party has no direct access to the tracker, the centralized architecture of BitTorrent becomes tedious: As the tracker only responds with a limited number of random peers in the swarm (usually 20), capturing a complete image of all nodes can only be approached by sequentially querying the tracker multiple times in order to improve the coverage. Due to the nature of this technique, capturing the whole network can only be achieved with a given level of confidence.
- d) Real participation: A very intuitive method of measuring P2P activity is for the monitoring party to participate in the network as a user. Erman et al. [7] have done first measurements by simply modifying the BitTorrent reference client to write log files of all communication. These logs can then later be used to extract a variety of data. The downside of this approach however, is that the participating client is controlled by the monitoring party and has to simulate the behaviour of an actual user. If a subset of real users could be supplied with the modified (monitoring) client, this technique could be described as sampling. Besides the obvious problem of the sample of such a user set adding a bias to the observed data, this approach is impractical as it requires cooperation by end-users who have no incentive to participate in such studies. Because the nodes under examination are regarded as an intrinsic part of the system, this technique is categorized as passive.

TABLE II
OVERVIEW OF MONITORING TOOLS

Tool	Type	System
Mistral [24]	Monitoring nodes	KAD
Montra [16]	Crawler and MVMs	KAD (eMule)
Blizzard [27]	Crawler	KAD
Rememj [32]	Crawler and monitoring node	KAD
Cruiser [28]	Crawler	other (Gnutella)
SkyEye.KOM [9]	Monitoring overlay system	any DHT
SOMO [33]	Monitoring overlay system	any DHT

e) Monitoring overlay: The concept of enhancing clients with monitoring capabilities has been taken further by Graffi et al. [9] with their monitoring software SkyEye.KOM. This tool runs on top of an existing overlay of a structured P2P system and creates an over-overlay of nodes participating in the measuring operations, effectively creating an Information Management Layer. As it is built on top of an existing overlay structure, costs for maintaining the over-overlay are very low, while it has good query performance. We will see in Section VI that measurement systems like SkyEye, which are themselves distributed, lead to a further distinction of the aggregation method of the collected data.

B. Active monitoring techniques

a) Crawling: This technique is restricted to systems in which peers can be introspected remotely. For example it is possible to query a peer for its neighbours in KAD networks. This is the major requirement for a crawl to be applicable. A Crawler walks the entire graph of the network by iteratively exploring the neighbours of known nodes. The algorithm to chose the order in which to visit the nodes is crucial to the effectiveness of the crawl: When simply selecting a random node from the set of all unexplored neighbours, the crawl would add a bias toward nodes with a high peer degree, as their neighbours are more likely to be selected. In order to prevent this, the Metropolized Random Walk is used to select each peer with equal probability [30].

Another main challenge for crawling is distortion: A single crawl merely attempts to take a *snapshot* of the network topology at one point in time. As the crawl itself takes time, this snapshot becomes distorted because the topology is already changing while the crawl takes place. By analyzing subsequent crawls, a sequence of snapshots can be used to infer dynamic behaviour over time (such as a peers session length). The "resolution" of this data is obviously upper-bounded by the time required to perform a single crawl. Furthermore, some properties might not be obtainable at all by a crawling: A peer might not provide information such as their transaction details; what objects they have been requesting and/or providing and what routing tasks have taken place. None of the studied protocols enable such detailed introspection of peers.

A common variation of a full crawl is a *partial crawl*, which restricts the iteration to a subset of nodes. Many studies [23] [27] [21] restrict their crawling to a part of the network to reduce crawling time. In Kademlia this can easily be implemented by choosing to only accept peers with shared prefix of $KAD\ ID$ bits [23]. For a prefix of n bits the subset

TABLE III
ADVANTAGES AND DISADVANTAGES OF MEASUREMENT TECHNIQUES

Technique	⊕ Advantages	⊕ Disadvantages
Traffic interception	perfect invisibility perfect communication coverage	 requires access to gateway high computational costs only captures activity, not inactive nodes
Tracker log analysis	 no additional real-time computational effort perfect invisibility perfect coverage of all nodes 	 restricted to BitTorrent requires access to tracker monitorable properties are very limited
Real participation	complete coverage of client activity for peer subset no network distortion	selection of peers adds biasdifficult to distribute to users
Crawling	no network disruption when lacking a tracker necessary for node discovery (KAD)	high traffic costs can only capture approximated snapshots
Monitoring nodes	does not require cooperation by peers, access to gateway points, or a centralized tracker	 only a fraction of the traffic can be captured reconstructed data is an approximation monitors add disruption to the network
Monitoring overlay	highly efficient through non-centralized data aggregation	deployment of monitoring peers difficult aggregated data is only an approximation

is then a fraction of $\frac{1}{n+1}$ of the network.

b) Measurement nodes: A different approach, applying to KAD networks, is to add peers simply for the purpose of capturing messages. These measurement nodes logically participate in the network, but do not act as real users themselves. Such a measurement node effectively only captures information in close proximity to its own KAD ID. Therefore, many of such nodes are required to capture a large fraction of the messages in regard to a certain zone. The motivation for using these nodes is their feasibility. Simply placing peers in the network neither requires the cooperation of end-users, as for real participation, nor does it require access to gateways for monitoring all network traffic, nor does it require access to a centralized service such as a tracker. Due to these restrictions, this technique is often the only possibility to make measurements. The setback with this approach is, that a high density of measurement nodes distorts the image of the real network. Qiao et al. [18] have placed only four measurement nodes in a KAD-based network to capture traffic. However, the coverage of peers observed this way is relatively small. Steiner et al. [24] have developed a tool called *Mistral* which places a large number of monitors into a subzone of a KAD network. However, this approach increases the natural density of nodes by eight times [16] and routes incoming traffic only to its own monitors. This disrupts the natural flow of traffic and arguable provides an incorrect image of the network.

Memon et al. [16] introduce the concept of *Minimally Visible Monitors* (*MMVs*) in their monitoring tool called *Montra*. MMVs are monitoring nodes that aim to reduce the disruption of the network by only monitoring one target node. The MVM is placed next to the target by choosing a *KAD ID* as close as possible. This monitor reveals its existence only to the

target node by ignoring messages by all other nodes from the network. ⁸ In respect to the protocol, other peers will consider the MVM to be offline due to the lack of a response, and its influence on the network is minimized.

A requirement for techniques employing *measurement nodes* is the ability to freely chose the *KAD ID* when joining the network. This enables the monitoring party to restrict their monitors to a subzone of the ID space, as well as accurately placing a monitor next to an existing ID. In case of [16], this is an already participating node.

Another example of exploiting this freedom is to place monitors next to the hash of an existing object: Steiner et al. [25] also created remote nodes called *Sybils* that can be placed at arbitrary positions in the network. These peers not only monitor traffic, they can also execute a wide range of attacks, such as eclipsing (hiding) certain content, isolating peers from the network, and Distributed-Denial-of-Service attacks against the whole system. For this reason it is debatable if the free selection of a *KAD ID* is after all a desirable property, albeit useful to a wide range of measuring approaches.

Table III summarizes the advantages and disadvantages of the techniques discussed in this section. Table II gives an overview of some monitoring tools which have been developed, and lists their type.

VI. AGGREGATION TECHNIQUES

As we have seen in the previous section, the tools used for measurement are themselves distributed in nature; either by modifying the actual nodes participating in the network,

⁸In order to gather more meta-data about the shared content this restriction is lifted in an extension, resulting in slight disruption

or adding a number of monitoring peers to the network. For such distributed measurement systems, a design choice of data aggregation of the monitored data has to be made.

- a) Centralized collector: The data is gathered by a centralized collector and all monitoring nodes communicate with a single service which collects all results. The main advantage of this method is the availability of all data at one point. The monitoring nodes also don't have any access to the aggregated data, which might be desirable. The downside is the bottleneck caused by having a single service communicate with many monitors. A feasible number of monitoring nodes is likely to be limited to by this single point.
- b) Gossip-based aggregation: The monitoring nodes communicate with each other by forming an overlay themselves and aggregating the monitored data among all monitors equally. Since there is no need for a centralized collector, this technique has the benefit to scale well with increasing number of monitors. Jelasity et al. [12] have created a gossip-based aggregation protocol, providing all nodes of a network with an up-to-date estimate of the data held by all nodes. They have also created *T-Man* [11], a protocol to create a topology between such nodes for the purpose of data aggregation. This protocol selects appropriate neighbours for each node according to a ranking function. In case of systems like KAD networks, the goal is to minimize the distance between connected nodes in respect to the XOR metric. Kempe et al. [13] show that their algorithm for gossip-based aggregation makes the estimate converge to the true value exponentially
- c) Tree-based aggregation: A third option of data aggregation, as employed by SkyEye [9] and SOMO [33], a self-organizing meta-overlay for tree-based aggregation of data created by Zheng et al. A tree of fixed depth is created as an overlay, with its leaves being monitoring notes collecting the data. The data is then gathered from the bottom and propagates toward the root of the tree. This technique combines the efficiency of the gossip-based approach with the benefit of centralized data availability.

VII. EVALUATION

Despite the numerous projects aiming at monitoring P2P networks, not all of these techniques have been adequately evaluated by their authors. In order to evaluate and compare different approaches, it is necessary to first define metrics to make judgements about the monitoring system.

A. Metrics

Although it is desirable to have universal metrics which apply to all techniques, not all metrics can be applied to all measurement techniques. For example, as a crawler performs a different task than monitoring peers, its performance has to be evaluated differently. Table IV gives an overview of the applicability of these metrics to the most important measurement techniques.

a) Accuracy: The percentage of actual messages that have been captured.

TABLE IV
APPLICABILITY OF MEASUREMENT METRICS

Metric	Crawling	Monitoring	Traffic	Tracker
		nodes	intercep-	log
			tion	analysis
Accuracy	X	X		
Precision	X	X	X	X
Traffic cost	X	X		
Computational cost	X	X	X	
Invisibility	X	X		
Edge distortion	X			
Node distortion	X			

- b) Precision: The precision of message sent/received times that were recorded.
- c) Traffic cost: The amount of additional traffic caused for measurement purposes.
- d) Computational cost: The computational power required per amount of information.
- e) Invisibility: The transparency of the monitoring system toward regular peers in the system.
- f) Edge distortion: For snapshot-based monitoring techniques, all edges are checked for bi-directionality. If a connection between peers is only observed in one direction, this is an indicator of inaccuracy of the snapshot.
- g) Node distortion: For snapshot-based monitoring techniques, two consecutive snapshots are compared by their symmetric difference. If the snapshots were instantly fast, they would be identical and the distortion metric would be 0. Higher values quantify the inaccuracy introduced by the time it takes to perform a snapshot.

B. Evaluation Techniques

Calculating the introduced metrics for a given monitoring system is not a trivial task. Metrics like *node distortion* and *edge distortion* are intrinsic to snapshot-based monitoring systems and can be computed without additional effort.

For metrics like *accuracy* there is a need for a reference data set against which the gathered data has to be compared. If there exists a second system which can be considered more accurate, the *relative accuracy* could be calculated, by assuming the more reliable system to be correct.

A method of calculating accuracy without such a reference is the usage of *instrumented peers*. For this method, a set of peers are placed in the network during a monitoring process. These peers simulate real users and send out queries into the network, but also log all their activity. The percentage of these messages captured by the monitoring system, can then be used to infer general accuracy of the monitor [8].

VIII. CONCLUSIONS

Measurements in P2P systems have become increasingly important in the recent years. In this paper we motivated the need for measurements, such as the awareness of system behaviour for development purposes. A wide range of measurement techniques have been introduced and categorized. For measurement systems which are, in turn, of distributed nature, the alternatives for data aggregation have been discussed.

Measurement techniques are highly dependant on the implementation details of the monitored system. Based on these observations, it is likely that in case of a new P2P system emerging, it will necessitate new measurement approaches as well.

Throughout the studies observed in this paper, there is a large intersection of what data is actually being monitored. The approaches in gathering this data are fairly different, and the actual techniques of obtaining data are often not the primary focus of the study but rather the interpretation of its results. Consequently, the monitor itself is not always evaluated.

A set of metrics for evaluation of such techniques have been defined, and their applicability has been discussed.

REFERENCES

- [1] Chord protocol: http://pdos.csail.mit.edu/chord/. II-A
- [2] eMule Client: http://www.emule.com/. II-B
- [3] Khashmir: http://khashmir.sourceforge.net/. II-C
- [4] Pastry project: http://www.freepastry.org/. II-A
- [5] Vuze Bittorrent Client: http://www.vuze.com. II-C
- [6] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Monitoring and Controlling Content Access in KAD. In *International Conference* on Communications - ICC 2010, Capetown, South Africa, May 2010. IEEE. III-A
- [7] Adrian Popescu Arne A. Nilsson David Erman, Dragos Ilie. Measurement and analysis of bittorrent signaling traffic. 2004. I, V-A0d
- [8] Jarret Falkner, Michael Piatek, John P. John, Arvind Krishnamurthy, and Thomas Anderson. Profiling a million user dht. In *Proceedings of the* 7th ACM SIGCOMM conference on Internet measurement, IMC '07, pages 129–134, New York, NY, USA, 2007. ACM. IV-B0b, I, VII-B
- [9] Kalman Graffi, Aleksandra Kovacevic, Song Xiao, and Ralf Steinmetz. Skyeye.kom: An information management over-overlay for getting the oracle view on structured p2p systems. In *Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems*, pages 279–286, Washington, DC, USA, 2008. IEEE Computer Society. I. II. V-A0e. VI-0c
- [10] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proceedings of the* nineteenth ACM symposium on Operating systems principles, SOSP '03, pages 314–329, New York, NY, USA, 2003. ACM. I, IV-B0d, V-A0a
- [11] Mark Jelasity and Ozalp Babaoglu. T-man: Gossip-based overlay topology management. In In 3rd Int. Workshop on Engineering Self-Organising Applications, pages 1–15. Springer-Verlag, 2005. VI-0b
- [12] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. volume 23, pages 219–252, New York, NY, USA, August 2005. ACM. VI-0b
- [13] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, pages 482–, Washington, DC, USA, 2003. IEEE Computer Society. VI-0b
- [14] Kendy Kutzner and Thomas Fuhrmann. Measuring large overlay networks - the overnet example. In *In Proceedings of the 14th KiVS*, page 2005, 2005. II-B
- [15] Petar Maymounkov and David Mazis. Kademlia: A peer-to-peer information system based on the xor metric, 2002. II-B
- [16] Ghulam Memon, Reza Rejaie, Yang Guo, and Daniel Stutzbach. Large-scale monitoring of dht traffic. In *Proceedings of the 8th international conference on Peer-to-peer systems*, IPTPS'09, pages 11–11, Berkeley, CA, USA, 2009. USENIX Association. I, II, V-B0b
- [17] J. A. Pouwelse, P. Garbacki, D.H.J. Epema, and H. J. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In 4TH INTERNATIONAL WORKSHOP ON PEER-TO-PEER SYSTEMS (IPTPS) 2005.
- [18] Yi Qiao and Fabian E. Bustamante. Structured and unstructured overlays under the microscope - a measurement-based view of two p2p systems that people use. In *In Proceedings of the USENIX Annual Technical Conference*, 2006. I, V-B0b
- [19] Sandeep Sarat and Andreas Terzis. Measuring the storm worm network.
 II-B

- [20] Stefan Saroiu, Krishna P. Gummadi, and Steven D. Gribble. Measuring and analyzing the characteristics of napster and gnutella hosts. *Multi-media Syst.*, 9:170–184, August 2003. IV-A0c
- [21] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems. 2002. V-B0a
- [22] Hendrik Schulze and Klaus Mochalski. Internet study 2008/2009. Africa, pages 1–13, 2009. II-C
- [23] Moritz Steiner, Ernst W Biersack, and Taoufik En-Najjary. Actively monitoring peers in kad. In IPTPS'07, 6th International Workshop on Peer-to-Peer Systems, February 26-27, 2007, Bellevue, USA, 02 2007. V-R0a
- [24] Moritz Steiner, Wolfgang Effelsberg, and Taoufik En-najjary. Load reduction in the kad peer-to-peer system. In In Fifth International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P, 2007. II, V-B0b
- [25] Moritz Steiner, Taoufik En-najjary, and Ernst W. Biersack. Exploiting kad: Possible uses and misuses. ACM SIGCOMM CCR, 37:2007. V-B0b
- [26] Moritz Steiner, Taoufik En-Najjary, and Ernst W Biersack. Analyzing peer behavior in kad. Technical Report EURECOM+2358, Institut Eurecom, France, 10 2007. IV-A0d
- [27] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. A global view of kad. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 117–122, New York, NY, USA, 2007. ACM. I, II, V-B0a
- [28] Daniel Stutzbach and Reza Rejaie. Characterization of p2p systems. IV,
- [29] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 189–202, New York, NY, USA, 2006. ACM. IV-A0a, IV-A0b, IV-B0d
- [30] Daniel Stutzbach, Reza Rejaie, Nick Duffield, Subhabrata Sen, and Walter Willinger. On unbiased sampling for unstructured peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 27–40, New York, NY, USA, 2006. ACM. V-B0a
- [31] Ruben D. Torres, Mohammad Y. Hajjat, Sanjay G. Rao, Marco Mellia, and Maurizio M. Munafo. Inferring undesirable behavior from p2p traffic analysis. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, SIGMETRICS '09, pages 25–36, New York, NY, USA, 2009. ACM. III-A, I
- [32] Jie Yu, Liming Lu, Peng Xiao, Zhoujun Li, and Yuan Zhou. Monitoring, analyzing and characterizing lookup traffic in a large-scale dht. Comput. Commun., 34:1622–1629, August 2011. I, II
- [33] Zheng Zhang. Somo: Self-organized metadata overlay for resource management in p2p dht. In in P2P DHT. In Proceedings of IPTPS, 2003. II, VI-0c

Semantisches Routing in Peer-to-Peer Systemen

Benjamin Mohr

Zusammenfassung—In den letzten 10 Jahren vollzogen Informations- und Kommunikationsinfrastrukturen einen rasanten und extremen Dezentralisierungsprozess durch den P2P-Netzwerke mit riesigen Datenmengen entstanden sind, welche auf effiziente und effektive Suchmechanismen angewiesen sind. Um den steigenden Anforderungen in Bezug auf Skalierbarkeit und Effizienz solcher Suchmechanismen gerecht zu werden, sind Ansätze aus dem Semantischen Web mit P2P-Systemen kombiniert worden. Hierfür werden im weiteren Verlauf dieser Arbeit verschiedene semantische Routing-Technologien erläutert, diese anhand einiger Gemeinsamkeiten kategorisiert und in Relation zu traditionellen Systemen evaluiert. Ausgehend von der Evaluation zeigt diese Arbeit, dass für bestimmte Anwendungsgebiete eine Kombination mit semantischen Beziehungen die Effizienz verschiedener P2P-Systeme deutlich steigern kann.

Index Terms—semantic routing, query routing, peer-to-peer, peer selection.

I. EINLEITUNG

Peer-to-Peer(P2P)-Netzwerke ermöglichen ihren Anwendern auf der ganzen Welt verteilte Ressourcen direkt miteinander zu teilen. Die Anzahl der Nutzer der einzelnen P2P-Netzwerke und deren jeweilige Popularität steigen dabei immer weiter. Eines der Kernprobleme hierbei ist die Bereitstellung von effizienten und skalierbaren Suchfunktionen bei der Verwendung eines komplett dezentralisierten Netzwerks. Frühe P2P-Systeme wie BitTorrent [8] setzten hierbei für die Lokalisierung der einzelnen Ressourcen im Netzwerk häufig auf zentralisierte Protokolle.

Für das Suchen und Finden von Ressourcen in dezentralisierten, verteilten Netzwerken existieren verschiedene Ansätze. Die ersten Ansätze beruhten darauf, das Netzwerk mit Abfragen zu überfluten. Ein typisches P2P-Netzwerk, welches diesen Ansatz verwendete, war Gnutella [17]. Während dieser Ansatz gute Resultate auf die jeweiligen Abfragen lieferte, skalierte er mit einer steigenden Anzahl an Nutzern zunehmend schlechter. Um die Skalierbarkeit zu verbessern, setzen andere P2P-Systeme wie Kazaa [15] auf hybride Infrastrukturen. Hierbei werden sogenannte Super-Knoten eingeführt, welche als Index-Server für die anderen Peers agieren. Diese hybriden Systeme funktionieren relativ gut, jedoch haben hierbei mehrere Teilnehmer des Netzwerks die Funktion eines Servers, welcher verschiedene Risiken mit sich bringt.

Einen anderen Ansatz verfolgen verteilte Hash-Tabellen (engl. distributed hash table, DHT). Hierbei wird jeder Ressource mittels einer Hash-Funktion ein Schlüssel in einem linearen Wertebereich zugeordnet. Weiterhin ist jeder Knoten für einen Teilbereich dieses Wertebereichs zuständig, sodass der gesamte Wertebereich abgedeckt ist. Die Zuständigkeiten ändern sich dabei dynamisch. Durch diesen Ansatz können die einzelnen Schlüssel schnell nachgeschlagen werden, jedoch ist dies zum Suchen von Ressourcen eher ungeeignet. Wird

beispielsweise der Schlüssel zu *peer* nachgeschlagen, ergibt sich ein komplett anderer Hash-Wert als zu *peers*. Weiterhin entsteht durch die Zuordnung des Wertebereichs auf die einzelnen Peers häufig eine unnötige Belastung für das Netzwerk. Beispiele für solche DHT-basierten Ansätze sind Chord [23], CAN [22] oder Kademlia [20].

Im weiteren Verlauf dieser Arbeit wird der Ansatz des semantischen Routings näher erläutert. Im Gegensatz zu den zuvor vorgestellten Ansätzen, liegt der Fokus bei dem semantischen Routing auf den einzelnen Abfragen und nicht auf der Topologie des Netzwerks. So werden neue Abfragen häufig anhand gesammelter Informationen von bereits zuvor bearbeiteten Abfragen weitergeleitet. Die daraus resultierenden semantischen Relationen beziehen sich dabei auf das Wissen der jeweiligen Peers.

Die Gliederung dieser Arbeit ist wie folgt aufgebaut. In Kapitel 2 wird der Begriff des semantischen Routings eingeführt. Anschließend werden in Kapitel 3 verschiedene semantische Routing-Technologien dargestellt. Ausgehend von diesen Technologien wird in Kapitel 4 eine geeignete Einteilung anhand gemeinsamer Eigenschaften definiert und diese durch einige Beispiele erläutert. Kapitel 5 evaluiert in Relation zu traditionellen Routing-Ansätzen beispielhaft einige der zuvor vorgestellten Technologien. Abschließend wird in Kapitel 6 eine Zusammenfassung dieser Arbeit gegeben.

II. SEMANTISCHES ROUTING

Die zwei Hauptfunktionen von P2P-Systemen beschäftigen sich mit dem Finden von passenden Knoten für eine Abfrage und dem effizienten Routen der einzelnen Nachrichten im Netzwerk. Semantische Routing-Mechanismen spezialisieren sich hierbei eher auf Ersteres: dem Finden von passenden Knoten für eine gegebene Abfrage. Hierfür werden Ansätze aus dem Gebiet des Semantic Web genutzt. Das Semantische Web beschreibt ein neues Konzept für die Erweiterung des aktuellen Webs, damit Maschinen die von Menschen bereitgestellten Informationen besser verarbeiten können [2]. Hierfür werden den menschlichen Informationen eindeutige Beschreibungen ihrer Bedeutung in maschinenlesbarer Form zugeordnet. Ähnlich kann in verteilten Systemen dem Wissen und den Ressourcen der jeweiligen Peers eine semantische Bedeutung zugewiesen werden, sodass diese bei der Auswahl der passenden Knoten zu einer gegebenen Abfrage berücksichtigt werden können.

Die verschiedenen semantischen Bedeutungen, welche den einzelnen Peers zugewiesen werden, formen dabei häufig ein Overlay-Netz. Betrachtet man ein P2P-System zum Nachrichtenaustausch, bei dem sich die einzelnen Nachrichten in die drei Kategorien Wirtschaft, Sport und Kultur einteilen lassen, könnte sich zum Beispiel das in Abbildung 1 dargestellte Overlay-Netz bilden. Hierbei werden jedem Peer ein oder

mehrere Themen zugeordnet. Ausgehend von dieser Sicht des Netzwerks könnten Abfragen zu den verschiedenen Themen genau den Peers zugewiesen werden, welche Nachrichten zu dem jeweiligen Thema bereitstellen.

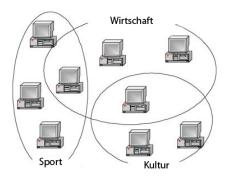


Abbildung 1. Beispiel eines semantischen Overlay-Netzes [9]

Ein Problem von traditionellen P2P-Systemen ist das Bereitstellen von möglichst vielen passenden Ergebnissen zu einer Abfrage in einer skalierbaren Art und Weise. Hierfür müssen die Abfragen an viele Peers weitergeleitet werden, während die Auslastung des Netzwerks möglichst gering bleibt. Um diese gegensätzlichen Ziele zu erreichen, können mit Hilfe von semantischen Beziehungen nur jene Peers befragt werden, die vermutlich die Abfrage beantworten können.

III. SEMANTISCHE ROUTING-TECHNOLOGIEN

Um die Effizienz und die Skalierbarkeit von aktuellen P2P-Systemen in Bezug auf das Suchen und Finden von verteilten Ressourcen weiter zu verbessern, sind Ansätze aus dem Bereich des semantischen Routings erforderlich. Im Folgenden werden verschiedene Technologien und Algorithmen erläutert, welche sich diesen Ansätzen bedienen.

A. P2P semantic link network

Ein solcher semantischer Ansatz wird in dem Artikel Query routing in a peer-to-peer semantic link network erläutert, bei dem der Begriff und die Eigenschaften eines semantisch verlinkten P2P-Netzwerks (P2P semantic link network, kurz P2PSLN) definiert werden [27]. Ein P2PSLN ist ein gerichtetes Netzwerk, bei dem jeder Knoten die Rolle eines Peers oder eines P2PSLN einnimmt. Jeder Knoten ist mit einer beliebigen Anzahl an anderen Knoten semantisch verbunden. Die semantischen Verbindungen entstehen dabei durch die jeweiligen XML-Schemata der einzelnen Peers, welche die im Netzwerk zu teilenden Ressourcen beschreiben. So können Informationen über die Ähnlichkeit verschiedener Knoten verwaltet werden. Eine Suchabfrage kann dann anhand dieser Informationen effizient an die relevanten Knoten des Netzwerks geroutet werden. Für die Kommunikation der Knoten untereinander werden hierbei auf SOAP [6] basierende Nachrichten verwendet. In dem Artikel wird dargestellt, dass ein solcher semantischer Ansatz für die Auswahl der relevanten Peers in Bezug auf Effizienz und Skalierbarkeit einen auf Zufall basierenden Ansatz übertrifft. Selbst im Vergleich zu einer kompletten Abfrage, bei der jeder Knoten des Netzwerks

befragt wird, ist die Effizienz nicht signifikant schlechter, jedoch die Skalierbarkeit deutlich besser.

B. Neurogrid

Neurogrid [16] stellt ein adaptives P2P-System dar. Jede erfolgreiche Suche verändert das Wissen eines jeden Peers, welcher an der Suche beteiligt ist. Hierfür verwaltet jeder Knoten des Netzwerks ein Wissensverzeichnis, welches verschiedenen Knoten Schlüsselwörter zuweist. Die Schlüsselwörter werden dabei aus den Meta-Daten der Dateien extrahiert. Jede Abfrage im Netzwerk wird anhand des jeweiligen Verzeichnisses weitergeleitet, indem das Verzeichnis nach geeigneten Knoten durchsucht wird. Knoten werden als geeignet gewertet, wenn sie möglichst viele identische oder ähnliche Schlüsselwörter zugewiesen bekommen haben. Das vorhandene Overlay-Netz passt sich ebenfalls adaptiv an. Jeder Knoten fügt nach einer erfolgreichen Suche den jeweiligen Knoten seiner Nachbarschaft hinzu, indem er sich die nötigen Informationen für eine direkte Verbindung speichert. Bei diesem semantischen Ansatz muss jeder Knoten eine enorme Menge an Verwaltungsinformationen speichern, weshalb dieser Ansatz eher ungeeignet

C. SWAP

SWAP (Semantic Web And Peer-to-Peer) bezeichnet eine dezentralisierte P2P-Plattform, welche sich semantischer Methoden zum Verteilen von Informationen und Dateien bedient [10]. Diese stellt Module zum Extrahieren von Informationen für lokale Peers, zum Speichern von Informationen und zum Routen von Abfragen bereit. Hierfür verwaltet jeder Peer ein lokales Knoten-Repository, welches die verschiedenen Informationen und Daten in Form von RDF(s)-Statements speichert [13]. Dabei können alle Peers selbst entscheiden, welche Daten dem eigenen Repository hinzugefügt werden. Zusätzlich zu dem eigenen Wissen eines jeden Peers werden in den Repositories noch Daten über andere Peers gespeichert. Die Statements im Repository können dabei von dem jeweiligen Peer bearbeitet und geändert oder durch erhaltene Nachrichten von anderen Peers beeinflusst werden. Den Abfragen der einzelnen Peers sind kaum Grenzen gesetzt. Diese können aus einfachen Konjunktionen oder komplexen Rekursionen, formuliert in einer RDF-basierten Abfragesprache, bestehen. Falls eine Abfrage nicht direkt beantwortet werden kann, wird diese in kleinere Abfragen aufgeteilt. Diese werden daraufhin im Netzwerk verteilt und die Antworten der verschiedenen Peers werden wieder zusammengeführt.

D. REMINDIN'

Der REMINDIN' (Routing Enabled by Memorizing INformation about Distributed INformation)-Algorithmus stellt einen auf sozialen Metaphern bezogenen Routing-Mechanismus bezüglich der Abfragen für die zuvor vorgestellte SWAP-Plattform bereit [24]. So würde ein Mensch, welcher nach Antworten auf eine Frage sucht, die folgenden Annahmen berücksichtigen:

1) Eine Frage wird der Person gestellt, von der man annimmt, diese am besten zu beantworten.

- 2) Eine Person gilt in einem bestimmten Gebiet als kompetent, wenn diese Person Fragen aus diesem Gebiet zuvor beantworten konnte.
- Falls eine Person in einem bestimmten Gebiet gut informiert ist, so ist sie dies wahrscheinlich auch in ähnlichen Gebieten.
- Jede Person ist unabhängig von dem aktuellen Thema mehr oder weniger kompetent.
- Die Kompetenz anderer Personen wird nicht anhand einer absoluten Skala, sondern anhand der eigenen Kompetenz gemessen.

REMINDIN' realisiert ein auf Metaphern aufbauendes P2P-Netzwerk, welches sich ähnlich wie ein menschliches soziales Netzwerk verhält, indem es die oben genannten Annahmen in einer algorithmischen Weise umsetzt. Dabei sucht dieses passende Knoten für Abfragen anhand einer Vertrauensbewertung, einer Lockerung der Abfrage und der Beobachtung von sinnvollen Antworten anderer Peers aus. Durch diesen Mechanismus stellt dieser Algorithmus signifikant bessere Resultate als ein naiver Ansatz wie Gnutella bereit.

E. INGA - semantisches Peer-to-Peer System

INGA beschreibt eine Strategie zum Auswählen von passenden Peers zu einer bestimmten Abfrage [19] ähnlich dem REMINDIN' Algorithmus. Bei INGA wird jedem Peer die Rolle eines Menschen in einem sozialen Netzwerk zugewiesen. Hierfür speichert und verwaltet jeder Peer seine Fakten lokal, welche seine aktuellen Informationen darstellen. Erhält ein Peer eine Abfrage, beantwortet er diese oder leitet diese zu aus seiner Sicht angemessenen Peers weiter. Für die Auswahl von angemessenen Peers verwaltet jeder Peer ein semantisches Verzeichnis. Dieses wird im Laufe der Zeit dynamisch gebildet, indem erhaltene und erstellte Abfragen analysiert werden. Weiterhin ordnet dieses Verzeichnis einem Peer in dem Kontext eines bestimmten Themas eine der folgenden vier Rollen zu:

- Inhaltsanbieter: Die geeignetsten Peers für eine Abfrage sind jene, welche bereits zuvor eine solche oder semantisch ähnliche Abfrage erfolgreich beantworten konnten.
- Empfehler: Falls keine Inhaltsanbieter bekannt sind, werden jene Peers befragt, welche zuvor semantisch ähnliche Abfragen erstellt haben.
- Bootstrapping-Netzwerk: Sind weder Inhaltsanbieter noch Empfehler bekannt, werden Peers befragt, welche eine große Auswahl an benachbarten Peers und somit verschiedenen Domänen haben.
- Default-Netzwerk: Falls keine Peers aus den oben genannten Bereichen bekannt sind, werden bekannte Peers aus der Nachbarschaft zufällig ausgewählt.

Lokal betrachtet hat jeder Peer durch sein Verzeichnis Kenntnis von anderen Peers, welche Rolle diese zu einem bestimmten Thema besitzen und wie nützlich diese bereits waren. Global betrachtet jedoch formt jede der oben genannten Rollen eine Netzwerkschicht, welche unabhängig von den anderen Schichten ist.

Weiterhin zeigen die Autoren von [19], dass dieser Ansatz mittels der SWAP-Plattform umgesetzt ähnliche Ergebnisse in

Bezug auf Effizienz und Skalierbarkeit wie der REMINDIN' Ansatz erzielt, diesen jedoch bei zunehmender Anzahl an Themen übertrifft.

F. Intelligent Search Mechanism

Bei dem Intelligent Search Mechanism (ISM) liegt der Fokus eher auf dem schnellen und effizienten Finden von relevanten Informationen, als auf dem Finden von vielen Informationen [26]. Hierbei werden Abfragen zu jenen Knoten weitergereicht, welche bereits zuvor ähnliche Abfragen beantworten konnten. Hierfür verwaltet jeder Knoten eine Liste mit Profilen von bekannten Knoten. Diese Profile protokollieren dabei, welche Abfragen der jeweilige Knoten beantworten konnte. Erhält ein Knoten nun eine Abfrage, kann dieser anhand der einzelnen Profile abschätzen, welche Knoten am ehesten diese Abfrage beantworten können und leitet die Abfrage dann ausschließlich an diese weiter. Eine Abfrage besteht hierbei aus verschiedenen Schlüsselwörtern, welche mit den Meta-Daten der verschiedenen Ressourcen der jeweiligen Knoten verglichen werden. Um die Ähnlichkeit zweier Abfragen festzustellen, können verschiedene Metriken wie die Kosinus-Ähnlichkeit verwendet werden [1]. Dabei verwendet der ISM ein vollständig verteiltes Netzwerk, bei dem jeder Knoten lokale und autonome Entscheidungen trifft, ohne sich mit anderen Peers koordinieren zu müssen.

G. Seers-Suchprotokoll

Das Suchprotokoll von Seers verbessert semantisches Routing mit Hilfe eines auf den Interessen der verschiedenen Peers aufbauenden Overlay-Netzes [5]. Peers mit ähnlichen Interessen sind im Overlay-Netz näher beieinander als Peers mit eher verschiedenen Interessen. Die Infrastruktur zum Suchen von relevanten Ressourcen basiert dabei auf sogenannten Meta-Dokumenten. Meta-Dokumente sind Beschreibungen für Ressourcen bestehend aus Meta-Daten dargestellt im XML-Format. Jede Suchanfrage an das System wird als Meta-Dokument repräsentiert, welches die gewünschte Ressource beschreibt. Jede Antwort auf die Anfrage wiederum wird als Meta-Dokument repräsentiert, welches eine existierende Ressource beschreibt und zu der erhaltenen Anfrage passt. Jede Applikation kann das Meta-Dokument erweitern, um die gewünschte oder eigene Ressource genauer zu beschreiben. Hierbei können sowohl die charakterisierenden Meta-Daten als auch die Regeln des Meta-Dokuments verändert werden. Dabei muss jedes Meta-Dokument Regeln bereitstellen, die bestimmen, wann zwei Meta-Daten von verschiedenen Dokumenten aufeinander abgebildet werden, wie das jeweilige Meta-Dokument zu übertragen ist und wie dieses gespeichert und weitergeleitet werden soll. Um ein plötzliches Auftreten von Antworten zu dem Initiator der Anfrage zu limitieren, werden alle Antworten über das Overlay-Netz zurück geroutet. Ausgehend von den erhaltenen Antworten zu seiner Anfrage, kann der Initiator dieser dann die jeweiligen Ressourcen mittels einer HTTP-Verbindung anfordern.

Weiterhin wurde das Seers-Suchprotokoll erweitert, sodass Präferenzen und Reputationen bei dem Routen von Anfragen berücksichtigt werden [4]. So werden bei der Auswahl an Nachbarn während der Weiterleitung von Anfragen zusätzlich zu dem Zustand und den Interessen dieser noch die Präferenzund Reputationsinformationen beachtet. Eine Präferenz ist hierbei eine lokal festgelegte Bewertung der Nachbarknoten basierend auf gesammelten Statistiken und Benutzereingaben. Eine Reputation dagegen ist ein asynchroner Austausch zwischen den verschiedenen Nachbarn über deren Verhalten.

H. SenPeer

SenPeer ist ein unstrukturiertes P2P-System, bei dem die einzelnen Peers anhand ihrer semantischen Domäne sogenannten Super-Peers zugeordnet werden [11]. Das Wissen der Peers wird dabei durch ein semantisches Netz, dem sGraph, dargestellt, welches verschiedene Datenmodelle einheitlich darstellen kann. Die Super-Peers verwalten Expertise-Tabellen, welche die Daten der semantisch verbundenen Peers und Super-Peers speichert, und definieren semantische Zuordnungen zwischen deren Inhaltsbeschreibungen. Diese Zuordnungen stellen den Kern des semantischen Overlay-Netzes dar, bei dem Peers mit ähnlichen Schemata nah beieinander sind und eine semantische Gruppe bilden. Anhand dieses Netzes werden die Suchanfragen der einzelnen Peers sinnvoll verteilt. Mit passenden Wrappern können hierfür verschiedene Datenmodelle gleichzeitig verwendet werden.

Die einzelnen Peers machen ihren zu verteilenden Inhalt mit Hilfe ihres Wissens bekannt und finden aufgrund geeigneter Schema-Matching Mechanismen bestehende Peers. Das Wissen der einzelnen Peers und die semantischen Zuordnungen organisieren so das Netzwerk in semantische Domänen. Durch die Einführung von Super-Peers mit ihren Expertise-Tabellen und dem Routing anhand der Domänen wird die Arbeitsbelastung der einzelnen Peers stark reduziert, da die Abfragen im Netzwerk auf verschiedene Gruppen von Peers verteilt werden.

I. Grid Resource Discovery

In dem Artikel A scalable semantic routing architecture for grid resource discovery wird ein hierarchischer, semantischer Routing-Algorithmus für Grid-Netzwerke vorgestellt [18]. Hierbei werden sowohl die Ressourcen als auch die Abfragen mit Hilfe von RDF [13] repräsentiert. Das Prinzip des Algorithmus' ist, den Inhalt der Abfrage mit dem Wissen des Netzwerks zu kombinieren, um so geeignete Entscheidungen bezüglich des Routings treffen zu können. Dabei werden die einzelnen Teilnehmer des Netzwerks anhand ihrer gemeinsamen Interessen in unabhängige Cluster gruppiert, wobei die Teilnehmer des selben Clusters eine Baumstruktur bilden. Aufgrund dieser Gruppierung kann ein Großteil der Abfragen innerhalb des eigenen Clusters beantwortet werden. Um für andere Cluster bestimmte Abfragen beantworten zu können, formen die Wurzelknoten der einzelnen Cluster ein Overlay-Netzwerk, sodass diese Abfragen effizient an die passenden Cluster verteilt werden können.

Zum Sammeln und Verwalten des Wissens des Netzwerks werden die jeweiligen Ressourcen in Form von RDF-Statements in einer effizienten Bitmap mit Hilfe von Bloomfiltern gespeichert [3]. Bloomfilter sind probabilistische Datenstrukturen zum Feststellen von neuen Daten in einem

kontinuierlichen Datenstrom. Hierfür wird durch verschiedene Hash-Funktionen ein "Fingerabdruck" der bereits bekannten Daten in einer Hash-Tabelle notiert. Die Baumstruktur der einzelnen Knoten eines Clusters beruht dabei auf die in [21] vorgestellte Methode. Hierbei stellt jeder neue Teilnehmer des Netzwerks, nachdem dieser in die jeweilige Baumstruktur eingefügt worden ist, seinem zugeordneten Vaterknoten eine Zusammenfassung seines Wissens in Form einer Bitmap bereit. So verwaltet jeder Knoten verschiedene Bitmaps über sein eigenes Wissen und dessen seiner Kindknoten. Erhält ein solcher Knoten eine neue Bitmap eines seiner Kindknoten, aggregiert er all seine Bitmaps und leitet sie ebenfalls zu seinem Vaterknoten weiter. Durch diese Methode hat jeder Knoten Kenntnis über das gesamte Wissen seiner Unterbäume und der Wurzelknoten über das gesamte Wissen des jeweiligen Clusters.

J. Directed Breadth First Search

Bei der *Directed Breadth First Search* Technik werden die einzelnen Abfragen zu einer ausgewählten Menge an Knoten, basierend auf den Erfahrungen bereits zuvor gestellten Abfragen, weitergeleitet [25]. Hierfür notiert jeder Knoten zu all seinen Nachbarn wie viele Ergebnisse er zu vorherigen Abfragen lieferte, wie viele Hops dieser brauchte um zu seinen Ergebnissen zu kommen, wie viele Abfragen er insgesamt weitergeleitet hat und wie lange seine Warteschlange für Abfragen ist. Anhand dieser Statistiken leitet er dann die jeweilige Abfrage an die geeignetsten Knoten weiter. Die Menge der Nachbarn eines Knoten verändert sich dabei nicht. Ein großer Nachteil hierbei ist, dass die einzelnen Ergebnisse nicht bewertet werden. So fließen auch falsche Ergebnisse in die Statistik mit ein.

IV. KATEGORISIERUNG SEMANTISCHER ROUTING-TECHNOLOGIEN

Semantische Routing-Mechanismen haben das Ziel, die Effektivität der Suchfunktionen zu maximieren während der Netzwerkverkehr auf ein Minimum reduziert wird. Hierfür existieren verschiedene Ansätze. Diese werden im weiteren Verlauf dieses Abschnittes analysiert und ausgehend von ihren Unterschieden im Kontext des semantischen Routings kategorisiert. Diese Analyse baut dabei auf die zuvor vorgestellten Ansätze auf, bei denen die autonomen Peers in Relation zu einander gesetzt worden sind, um das Wissen der einzelnen und unabhängigen Peers darstellen zu können. Ausgehend von diesem Aspekt, lassen sich die Ansätze dabei in Bezug zu ihrer Darstellung des Wissens der Peers in Meta-Daten basierte und Ontologie basierte Ansätze einteilen.

A. Meta-Daten basierte Technologien

Bei den auf Meta-Daten basierenden semantischen Routing-Technologien wird das Wissen eines jeden Knotens implizit definiert, indem dieser dem System verschiedene Ressourcen mit den dazugehörigen Meta-Daten bereitstellt. So lassen sich die einzelnen Peers aufgrund der verschiedenen Informationen, welche mit den geteilten Ressourcen verbunden werden, in Relation zueinander setzen. Um hierbei ähnliche oder für eine beliebige Abfrage passende Knoten erkennen zu können, existieren verschiedene Matching-Techniken. Hierbei werden die Meta-Daten oder Schlüsselwörter der jeweiligen Peers miteinander oder mit den aktuellen Abfragen verglichen. Die verschiedenen Technologien werden dabei durch den Routing-Mechanismus charakterisiert. Je nachdem wie sehr sich der Routing-Mechanismus an den Meta-Daten orientiert und wie diese Meta-Daten ausgewertet und verknüpft werden, lassen sich die einzelnen Systeme, welche dieser Kategorie zuzuordnen sind, unterscheiden.

Ein simples Beispiel für diese Art der Wissensdarstellung eines Peers ist das dezentralisierte P2P-System Neurogrid [16]. Hierbei extrahieren die jeweiligen Peers verschiedene Schlüsselwörter aus den zu teilenden Ressourcen und speichern diese in einer Liste. Weiterhin enthält jede Abfrage ebenfalls eine Menge an Schlüsselwörtern, anhand derer die relevanten Daten lokalisiert werden. Konnte ein Knoten des Netzwerks eine Abfrage beantworten, merken sich dies die anderen an der Suche beteiligten Knoten und weisen dem Knoten die jeweiligen Schlüsselwörter der Anfrage hinzu. Durch diesen Mechanismus passt sich das Netzwerk adaptiv an.

Ein komplexeres Beispiel aus dieser Kategorie stellt der *Intelligent Search Mechanism (ISM)* dar [26]. Hierbei basiert die Auswahl der passenden Knoten für eine Abfrage ebenfalls auf den Informationen, welche aus den zuvor gestellten Abfragen abgeleitet werden können. Jeder Knoten verwaltet dafür eine Liste von Profilen über die Knoten seiner Nachbarschaft. Eine Abfrage besteht hierbei aus Schlüsselwörtern, welche mit den Meta-Daten der Ressourcen verglichen werden. Anders als bei Neurogrid werden hier allerdings bei der Verteilung der Abfragen die nächsten Knoten nicht anhand einer genauen Übereinstimmung der Schlüsselwörter bestimmt, sondern anhand einer Ähnlichkeits-Metrik, wie zum Beispiel der Kosinus-Ähnlichkeit [1].

Ein anderes Beispiel stellt das Suchprotokoll von Seers dar [5]. Hierbei wird jede im Netzwerk geteilte Ressource und jede Abfrage anhand eines Meta-Dokuments im XML-Format dargestellt. Wann zwei Meta-Daten gleich bzw. ähnlich sind, wird dabei anhand der Meta-Dokumenten hinzugefügten Matching-Regeln bestimmt. Ebenfalls enthält jedes Meta-Dokument Regeln für die Art der Übertragung und Speicherung des jeweiligen Dokuments. Die Weiterleitung der jeweiligen Abfragen orientiert sich dabei auf die zuvor erhaltenen Meta-Dokumente der verschiedenen Peers.

B. Ontologie basierte Technologien

Bei den auf Ontologie basierenden semantischen Routing-Technologien wird das Wissen eines jeden Knotens explizit definiert, indem dieser dem System seine zu teilenden Ressourcen in Form einer Ontologie mitteilt. Diese verschiedenen Ontologien, welche die geteilten Ressourcen und damit das Wissen der einzelnen Peers beschreiben, bestimmen die Relation der jeweiligen Peers zueinander. Zusätzlich werden bei diesem Ansatz verschiedene Matching-Techniken definiert, um die verschiedenen Abfragen in Bezug zu den jeweiligen Ontologien semantisch auszuwerten. Anhand der spezifizierten

Ontologie und der Komplexität der Matching-Technik lassen sich dabei die verschiedenen Technologien, welche dieser Kategorie zuzuordnen sind, unterscheiden.

Ein simpler Ansatz Ontologie basierter Mechanismen wird in *Peer Selection in Peer-to-Peer Networks with Semantic Topologies* vorgestellt [14]. In diesem Artikel wird ein Auswahlverfahren für Peers erwähnt, das auf dem Veröffentlichen der jeweils eigenen Expertise bzw. dem Wissen der einzelnen Peers beruht. Die eigene Expertise wird dabei durch eine vom System vorgegebene Ontologie dargestellt. Die Kenntnis von den Expertisen anderer Peers formt dabei eine semantische Topologie, welche unabhängig von der darunter liegenden Netzwerk-Topologie ist. Erhält ein Peer eine Abfrage, leitet er diese an die Peers, deren Expertise dem Thema der Abfrage ähnelt, weiter. Hierfür verwaltet jeder Peer eine Liste von Peers, deren Expertise ähnlich der eigenen ist. Da jeder Knoten seine eigene Expertise veröffentlicht, sobald er das Netzwerk betritt, bildet sich so ein semantisches Overlay-Netz.

Ein komplexeres System, bei welchem das Wissen der einzelnen Knoten mittels einer Ontologie beschrieben wird, ist SWAP (Semantic Web And Peer-to-Peer) [10]. Hierbei verwaltet jeder Knoten ein lokales Repository, welches die einzelnen Daten und verschiedenen Informationen in Form von RDF(s)-Statements speichert [13]. Als Matching-Technik kann hierbei jede beliebige RQL-Sprache für die Formulierung der einzelnen Abfragen benutzt werden. Empfehlenswert ist hierfür die SeRQL-Sprache [7]. Erhält ein Knoten des SWAP-Netzwerks eine in RQL formulierte Abfrage, antwortet dieser mit allen passenden RDF(s)-Statements seines lokalen Repositories. Anhand dieser Antworten können die einzelnen Peers des Netzwerks jedem anderen bekannten Knoten einen Kompetenz-Wert zuordnen, welcher die Anzahl der relevanten Statements zu den jeweiligen Abfragen widerspiegelt.

REMINDIN' [10] und INGA [19], welche beide auf die SWAP-Plattform aufbauen, nutzen diesen Kompetenz-Wert dabei, um die Abfragen zu den geeignetsten Knoten weiterzuleiten. Beide erweitern dabei diesen Kompetenz-Wert, indem sie weitere Faktoren berücksichtigen. Diese Faktoren entsprechen dabei verschiedenen Eigenschaften aus sozialen Metaphern.

Ein weiteres Beispiel für Ontologie basierte Systeme ist das Peer Data Management System (PDMS) SenPeer [11]. Hierbei veröffentlichen die einzelnen Peers ihr Wissen bzw. Schema in Form eines Datenmodell-unabhängigen semantischen Netzes, dem sGraph, um so ihre zu verteilenden Daten bekannt zu geben. Ein sGraph ist dabei ein gerichteter Graph, dessen Knoten die einzelnen Elemente des vom Peer gewählten Schemas darstellen und dessen Kanten mögliche semantische Verbindungen zwischen diesen repräsentiert. Zusätzlich hat der jeweilige Nutzer die Möglichkeit Schlüsselwörter zu den einzelnen Knoten des sGraphs hinzuzufügen. Abbildung 2 zeigt einen Ausschnitt aus einem typischen sGraphen. Die Ähnlichkeit zweier sGraphen wird dann anhand der semantischen Bedeutung und der Schlüsselwörter der einzelnen Knoten gebildet [12].

Ausgehend von dieser Messung lassen sich die einzelnen Peers in semantische Domänen unterteilen. So können diese anhand ihrer jeweiligen semantischen Domäne einem

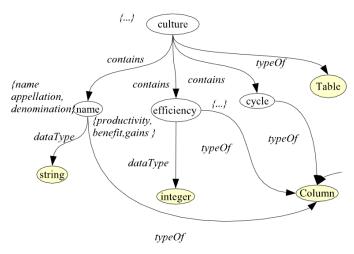


Abbildung 2. Ausschnitt aus einem sGraphen [11]

Super-Peer zugewiesen werden, woraus sich verschiedene semantische Nachbarschaften ergeben. Um später Abfragen effizient weiterleiten zu können, verwalten diese Super-Peers die einzelnen Informationen ihrer ihnen zugewiesenen Peers und anderen Super-Peers in sogenannten Expertise-Tabellen. Dadurch können die Inhalte der verschiedenen Peers semantisch verknüpft werden. Diese Verknüpfungen stellen dabei die Basis des semantischen Overlay-Netzes dar, welches das Netzwerk in semantische Domänen einteilt.

V. SEMANTISCHE ROUTING-TECHNOLOGIEN IM VERGLEICH

Der Fokus semantischer Routing-Mechanismen liegt vor allem auf skalierbaren und effizienten Suchverfahren. Im weiteren Verlauf wird anhand einiger Beispiele gezeigt, dass semantische Routing-Mechanismen in der Lage sind, traditionelle P2P-Systeme in diesem Gebiet zu übertreffen und somit für verschiedene Anwendungen von Vorteil sein können. Um diese unterschiedlichen Ansätze vergleichen zu können, werden die folgenden Metriken definiert:

- Trefferquote (engl. recall) ist die Wahrscheinlichkeit, mit der relevante Informationen und Daten gefunden werden. Diese beschreibt für eine Abfrage das Verhältnis zwischen allen relevanten Daten im Netzwerk und den erhaltenen Daten, womit sich ein Messwert für die Effizienz der Suchmechanismen ergibt.
- Nachrichten bezeichnet die Suchkosten für eine Abfrage. Diese kann als Messwert für die Skalierbarkeit des Systems verwendet werden.

In [19] zeigt der Autor, dass INGA in beiden Punkten ein traditionelles P2P-System wie Gnutella [17] schlägt. Da sich der Routing-Mechanismus bei INGA im Laufe der Zeit verbessert, ist bei wenigen Abfragen die Trefferquote noch niedriger als bei Gnutella. Nimmt die Zahl der Abfragen zu, verbessert sich die Trefferquote von INGA und weist dadurch eine deutliche höhere Quote als Gnutella auf. Weiterhin ist die Anzahl der benötigten Nachrichten für die einzelnen Abfragen deutlich niedriger als bei Gnutella. Diese Werte richten sich

jedoch nach der gegebenen Simulationsumgebung und können deshalb je nach Anwendung variieren.

Ebenfalls mit dem Gnutella-Protokoll wird der *Intelligent Search Mechanism(ISM)* in [26] verglichen. Hierbei ist die Trefferquote des ISM nach einigen Abfragen etwas niedriger als die des Gnutella-Protokolls, jedoch werden nur noch halb so viele Nachrichten benötigt. Die Unterschiede können dabei je nach Einstellungen und Umgebung variieren.

Ein weiteres Beispiel wird in [11] vorgestellt. Hierbei wird ein auf SenPeer aufbauendes System mit einem System verglichen, welches ähnlich wie Kazaa [15] die einzelnen Peers zufällig einem Super-Knoten zuweist und dieser die Abfragen ebenfalls zufällig an ihm bekannte Peers weiterleitet. Die in dem Artikel dargestellte Simulation zeigt, dass SenPeer weniger Nachrichten benötigt und eine höhere Trefferquote aufweist als das alternative System, wobei die Unterschiede mit zunehmender Anzahl an Peers immer größer werden.

VI. FAZIT

Dezentrale Peer-to-Peer-Netzwerke scheinen sich gut für die Verteilung von Informationen im Internet zu eignen, da die Vorteile in Bezug auf Zuverlässigkeit und Skalierbarkeit überzeugend sind. Hierbei werden effektive und effiziente Suchalgorithmen durch eine steigende Anzahl an Daten immer wichtiger. Naive Ansätze für die Organisation eines solchen Netzwerks brechen jedoch schnell zusammen, wodurch die eben genannten Vorteile zu Schwächen werden. Unter diesem Aspekt wurde ein semantischer Ansatz definiert und verschiedene Systeme, welche sich darauf konzentrieren, vorgestellt. Aufgrund der verschiedenen Herausforderungen in Bezug auf ein vollständig verteiltes Netzwerk mit autonomen, individuellen Peers konzentrierten sich die einzelnen Technologien dabei auf verschiedene semantische Schwerpunkte.

Weiterhin wurden essenzielle Gemeinsamkeiten der einzelnen Systeme aufgezeigt, aus denen sich eine mögliche Einteilung in semantische Routing-Mechanismen ergibt. Hierbei wurden auch verschiedene Charakteristiken der jeweiligen Gemeinsamkeit dargestellt und an einigen Technologien erläutert. Diese Charakteristiken sind dabei ein Indiz auf die Komplexität der jeweiligen Technologie.

Zum Schluss wurden verschiedene semantische Routing-Mechanismen anhand naiver Technologien evaluiert. Hierbei wurde gezeigt, dass verschiedene semantische Ansätze eine deutlich bessere Effizienz erreichen können als einfache P2P-Systeme ohne entsprechende semantische Beziehungen. Es lässt sich allerdings keine allgemeingültige Aussage treffen, ob sich ein semantischer Ansatz lohnt und welche Systeme man nutzen sollte. Dies richtet sich immer an den gegebenen Kontext und der Umgebung der Anwendung. So sollte man nicht immer direkt semantische Technologien nutzen, sondern sich genau überlegen, ob ein semantischer Ansatz für die jeweilige Anwendungsumgebung rentabel ist.

LITERATUR

- [1] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1st edition, May 1999. III-F, IV-A
- [2] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. Scientific American, 284(5):34–43, 2001. II

- [3] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13:422–426, 1970. III-I
- [4] Njal T. Borch. Improving semantic routing efficiency. In Proceedings of the Second International Workshop on Hot Topics in Peer-to-Peer Systems, pages 80–86, 2005. III-G
- [5] Njal T. Borch and Lars Kristian Vognild. Searching in variably connected p2p networks. In Hamid R. Arabnia, Laurence Tianruo Yang, and Chi-Hsiang Yeh, editors, Proceedings of the International Conference on Wireless Networks, ICWN 04, Volume 2 & Proceedings of the International Conference on Pervasive Computing and Communications, PCC 04, June 21-24, 2004, Las Vegas, Nevada, USA, pages 806–812, 2004. III-G, IV-A
- [6] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple Object Access Protocol (SOAP) 1.1. W3c note, World Wide Web Consortium, May 2000. III-A
- [7] Jeen Broekstra, Marc Ehrig, Peter Haase, Frank Van Harmelen, Arjohn Kampman, Marta Sabou, Ronny Siebes, Steffen Staab, Heiner Stuckenschmidt, and Christoph Tempich. A metadata model for semantics-based peer-to-peer systems. In In: Proceedings of the WWW'03 Workshop on Semantics in Peer-to-Peer and Grid Computing, 2003. IV-B
- [8] Bram Cohen. Incentives build robustness in bittorrent, 2003. I
- [9] Christos Doulkeridis, Akrivi Vlachou, Kjetil Norvag, and Michalis Vazirgiannis. Distributed semantic overlay networks. In Xuemin Shen, Heather Yu, John Buford, and Mursalin Akon, editors, *Handbook of Peer-to-Peer Networking*, pages 463–494. 2010. 1
- [10] Marc Ehrig, Christoph Tempich, Jeen Broekstra, Frank van Harmelen, Marta Sabou, Ronny Siebes, Steffen Staab, and Heiner Stuckenschmidt. Swap: Ontology-based knowledge management with peer-to-peer technology, 2003. III-C, IV-B
- [11] David Faye, Gilles Nachouki, and Patrick Valduriez. Semantic query routing in senpeer, a p2p data management system. In *Proceedings of* the 1st international conference on Network-based information systems, NBiS'07, pages 365–374, 2007. III-H, IV-B, 2, V
- [12] David C. Faye, Gilles Nachouki, and Patrick Valduriez. SenPeer: un système pair-à-pair de médiation de données. ARIMA, 4:24–52, May 2006. IV-B
- [13] Ramanathan V. Guha and Dan Brickley. RDF Vocabulary Description Language 1.0: RDF Schema. W3C recommendation, W3C, February 2004. III-C, III-I, IV-B
- [14] Peter Haase, Ronny Siebes, and Frank Van Harmelen. F.: Peer selection in peer-to-peer networks with semantic topologies. In *In: International Conference on Semantics of a Networked World: Semantics for Grid Databases*, 2004. IV-B
- [15] Jian Liang Information and Jian Liang. Understanding kazaa, 2004. I,
- [16] Sam Joseph. Neurogrid: Semantically routing queries in peer-to-peer networks. In *In Proc. Intl. Workshop on Peer-to-Peer Computing*, pages 202–214, 2002. III-B, IV-A
- [17] Gene Kan. Gnutella. In Detlef Schoder, Kai Fischbach, and Rene Teichmann, editors, *Peer-to-Peer*, Xpert.press, pages 189–199. 2002. I,
- [18] J. Li and Son Vuong. A scalable semantic routing architecture for grid resource discovery. In *Parallel and Distributed Systems*, 2005. *Proceedings*. 11th International Conference on, volume 1, pages 29 – 35 Vol. 1, july 2005. III-I
- [19] Alexander Löser, Steffen Staab, and Christoph Tempich. Semantic methods for p2p query routing. In Torsten Eymann, Franziska Klügl, Winfried Lamersdorf, Matthias Klusch, and Michael Huhns, editors, Multiagent System Technologies, volume 3550 of Lecture Notes in Computer Science, pages 15–26. 2005. III-E, IV-B, V
- [20] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In Peter Druschel, Frans Kaashoek, and Antony Rowstron, editors, *Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, pages 53–65. 2002. I
- [21] Michael T. Prinkey. An efficient scheme for query processing on peerto-peer networks. 2002. III-I
- [22] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott 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, 2001. I
- [23] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications*, technologies, architectures, and protocols for computer communications, SIGCOMM '01, pages 149–160, 2001. I

- [24] Christoph Tempich, Steffen Staab, and Adrian Wranik. Remindin': Semantic query routing in peer-to-peer networks based on social metaphors, 2004. III-D
- [25] Beverly Yang and Hector Garcia-molina. Efficient search in peer-to-peer networks. 2002. III-J
- [26] D. Zeinalipour-yazti, Vana Kalogeraki, and Dimitrios Gunopulos. Exploiting locality for scalable information retrieval in peer-to-peer networks. *Information Systems*, 30:277–298, 2005. III-F, IV-A, V
- [27] Hai Zhuge, Jie Liu, Liang Feng, Xiaoping Sun, and Chao He. Query routing in a peer-to-peer semantic link network. *Computational Intelli*gence, 21:197–216, 2005. III-A

Failover und Switchover von Diensten und Anwendungen in verteilten Systemen

Sebastian Puttkammer

Zusammenfassung—Durch die steigende Komplexität aktueller verteilter Systeme und dem natürlichen Verschleiß von Hardware lässt sich ein Ausfall verschiedener Systemkomponenten nicht immer verhindern. Failover- und Switchoversysteme ermöglichen auch bei Ausfällen von einzelnen Komponenten eine zuverlässige Bereitstellung von Diensten und Anwendungen in verteilten Systemen. In dieser Arbeit werden die Grundkomponenten von Failover- und Switchoversystemen betrachtet und auf ihre Anwendbarkeit untersucht. Dabei zeigt sich, dass je nach Anwendung und deren Anforderungen unterschiedliche Ansätze und Techniken verfolgt werden müssen.

I. EINLEITUNG

Die zuverlässige Bereitstellung von Diensten und Anwendungen in verteilten Systemen, wie dem Internet, spielt heutzutage eine immer wichtigere Rolle. Fehlertolerante Systeme können verwendet werden, um die Verfügbarkeit und Zuverlässigkeit von Netzwerkdiensten und -Anwendungen zu erhöhen. Wichtige Aspekte solcher Systeme sind Failover und Switchover. Als Failover wird der ungeplante Wechsel zu einem Zusatz- oder Ersatzsystem, bei dem Ausfall des zuvor verwendetem Hauptsystems, bezeichnet. Auf diese Weise kann ein Dienst oder eine Anwendung auch bei einem Ausfall weiter zur Verfügung gestellt werden. Switchover hingegen bezeichnet einen geplanten Systemwechsel, der zum Beispiel von einem Systemadministrator eingeleitet werden kann. Dies wird üblicherweise durchgeführt, um Wartungsarbeiten am Hauptsystem durchführen zu können oder die Funktionsfähigkeit des Ersatzsystems zu überprüfen.

Ein Failover/Switchover-System muss je nach Anforderungen verschiedenste Aufgaben realisieren. Zum Einen ist es notwendig den aktuellen Dienst- oder Anwendungszustand zu sichern und diesen auf einem oder mehreren Ersatzsystemen bereitzustellen.

Tritt ein Switch- oder Failover auf, müssen neue Anfragen an den Dienst oder die Anwendung an einen Ersatzserver umgeleitet werden. Für Dienste, wie zum Beispiel Streaming-Anwendungen, steigt die Wichtigkeit für Dienst-Kontinuität, die ununterbrochende Bereitstellung eines Dienstes. Um diese zu gewährleisten, müssen bei einem Fail- oder Switchover auch bereits etablierte Verbindungen reibungslos von einem Ersatzsystem übernommen werden. Dadurch können Dienste, auch bei Auftreten von Fehlern, sitzungsübergreifend verlässlich angeboten werden. Damit dies möglich ist, müssen Ausfälle des Dienstes oder der Anwendung schnell und zuverlässig erkannt werden. Diese Aufgabe übernehmen in der Regel Fehlerdetektoren, die bei einem Ausfall den Failover initiieren.

Der Rest dieser Arbeit ist wie folgt organisiert: Der nächste Abschnitt gibt eine Übersicht über verschiedene Fehlerdetektoren, die wichtig für die Auslösung des Failovers sind. Im An-

schluss werden Techniken für die Sicherung und Wiederherstellung des Zustands einer Anwendung vorgestellt. Daraufhin werden verschiedene Möglichkeiten für einen Systemwechsel und die reibungslose Übernahme von bestehenden Verbindungen beschrieben. Zum Abschluss werden die Ergebnisse der Arbeit zusammengefasst.

II. FEHLERDETEKTOREN

Fehlerdetektoren sind ein Grundbaustein für fehlertolerante, verteilte Systeme. Mit Hilfe eines Fehlerdetektors können kritische Prozesse einer Anwendung überwacht werden. Sie liefern Informationen über den Status von Prozessen, üblicherweise in Form einer Liste von Prozessen, die verdächtigt werden abgestürzt zu sein. Diese Statusinformationen können jedoch nicht korrekt oder nicht auf dem aktuellen Stand sein. Dies ist durch die Verzögerung oder dem Verlust von Nachrichten bedingt.

Im Folgenden werden zunächst Anforderungen und Qualitätsmerkmale von Fehlerdetektoren vorgestellt, die es ermöglichen verschiedene Detektoren miteinander zu vergleichen. Daraufhin werden übliche Implementierungsstrategien und Funktionsweisen von Fehlerdetektoren vorgestellt und gegeneinander abgewägt.

A. Anforderungen

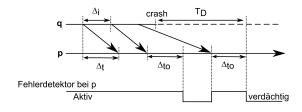
Wichtige Anforderungen an Fehlerdetektoren sind Vollständigkeit und Effizienz. Vollständigkeit bedeutet, dass ein abgestürzter Prozess letztendlich immer erkannt wird. Die Effizienz eines Fehlerdetektors hängt von der Geschwindigkeit der Fehlererkennung und der Genauigkeit ab. Die Genauigkeit beschreibt, wie gut vermieden wird, dass aktive Prozesse als abgestürzt identifiziert werden.

Um die *Effizienz* zu messen, können verschiedene Metriken verwendet werden. Die Geschwindigkeit eines Fehlerdetektors wird anhand der Zeit bestimmt, die zwischen dem Absturz eines Prozesses und dem Erkennen vergeht. Es existieren viele verschiedene Aspekte für die Genauigkeit, die je nach Anwendung des Detektors relevant sind. In Folge dessen, werden üblicherweise zwei nicht redundante Metriken verwendet, aus denen sich jedoch andere wichtige Metriken ableiten lassen: die Zeit zwischen dem Auftreten von zwei aufeinanderfolgenden Fehlabschätzungen und die Zeit, die notwendig ist, einen aufgetretenen Fehler zu korrigieren [21].

B. Strategien

Es existieren zwei klassische Implementierungsstrategien für Fehlerdetektoren: die *Heartbeat-Strategie* und die *Ping-Strategie*. Diese Strategien werden in Abbildung 1 illustriert.

Heartbeat-Strategie



Ping-Strategie

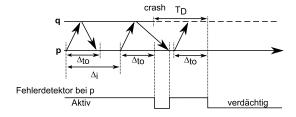


Abbildung 1. Heartbeat- und Ping-Strategie für die Fehlerdetektion

Es werden jeweils zwei Prozesse p und q betrachtet. Prozess p repräsentiert einen Fehlerdetektor, während q den überwachten Prozess darstellt. In dem gewählten Beispiel wird lediglich ein Prozess überwacht, jedoch kann diese Überwachung auf mehrere Prozesse ausgeweitet werden.

1) Heartbeat-Strategie: Bei der Heartbeat-Strategie sendet jeder überwachte Prozess q eine Nachricht, den sogenannten Heartbeat (Herzschlag), in Intervallen von Δ_i an einen Fehlerdetektor, in diesem Fall an p. Diese Nachricht signalisiert, dass der Prozess noch aktiv ist. Erhält der Fehlerdetektor für einen Zeitraum Δ_{TO} keine Nachricht von q, wird angenommen, dass der Prozess abgestürzt ist. Wenn eine Nachricht später eintrifft, wird der entsprechende Prozess aus der Liste der vermeintlich abgestürzten Prozesse wieder entfernt [10].

Die klassische *Heartbeat-Strategie* besitzt zwei unerwünschte Eigenschaften, die die Genauigkeit und die Detektionszeit negativ beeinflussen. Zum Einen ist die Fehlerdetektion immer von zwei Heartbeats abhängig. Wird ein Heartbeat schneller übertragen, startet das Intervall Δ_{TO} früher, wodurch die Wahrscheinlichkeit steigt, dass der nächste Heartbeat nicht mehr in dem vorgegebenen Intervall empfangen und der Prozess fälschlicherweise als abgestürzt angesehen wird. Dadurch sinkt die Genauigkeit, was nicht erwünscht ist. Zum Anderen ist die maximale Detektionszeit gegeben durch die maximale Übertragungszeit, addiert mit dem Intervall Δ_{TO} . Dadurch ist die Detektionszeit relativ hoch [21].

Um die angesprochenen Probleme zu beheben, wird die klassische *Heartbeat-Strategie* meist folgendermaßen angepasst: Der überwachte Prozess p sendet die Heartbeatnachrichten periodisch zu festen Zeitpunkten. Um festzustellen, ob der Prozess noch aktiv ist, verwendet der Prozess q sogenannte "freshness points". Diese werden errechnet, indem der Parameter Δ_{TO} mit dem Sendezeitpunkt σ_i eines Heartbeats m_i addiert wird: $\tau_i = \sigma_i + \Delta_{TO}$. Empfängt Prozess q eine Nachricht m_i oder eine spätere im Intervall $[\tau_i, \tau_{i+1}]$, wird der Prozess als aktiv angesehen, anderenfalls als abgestürzt. Der erläuterte Algorithmus benötigt synchronisierte Uhren, kann

jedoch durch die Verwendung von erwarteten Ankunftszeiten von Heartbeats so modifiziert werden, sodass diese nicht mehr notwendig sind, aber gleichwertige Ergebnisse liefert [21].

- 2) Ping-Strategie: Bei der Ping-Strategie fragt der Fehlerdetektor in regelmäßigen Abständen Δ_i den überwachten Prozess, ob dieser noch aktiv ist. Bei dem Erhalt einer solchen Nachricht, antwortet der Prozess entsprechend, wodurch seine Aktivität bestätigt wird. Empfängt Prozess p innerhalb des Timeout-Intervalls Δ_{TO} keine Bestätigung, wird der Prozess, wie in der Heartbeat-Strategie, zu der Liste der vermutlich abgestürzten Prozesse hinzugefügt und bei einem nachträglichen Erhalt wieder entfernt [10].
- 3) Vergleich der Strategien: Im Vergleich der beiden Strategien, besitzt die Heartbeat-Strategie einige Vorteile gegenüber der Ping-Strategie. Es werden zum Einen nur halb so viele Nachrichten, bei identischer Detektionsqualität, benötigt. Zum Anderen lässt sich die Timeout-Zeit Δ_{TO} leichter abschätzen. Bei der Heartbeat-Strategie muss nur die Übertragungsverzögerung eines Heartbeats geschätzt werden, während bei der Ping-Strategie die Übertragungszeit der Anfrage, die Verarbeitungszeit und die Übertragungszeit der Antwort kalkuliert werden müssen [10].

Die *Effizienz* eines auf diese Weise implementierten Fehlerdetektors hängt hauptsächlich von der Wahl der Parameter Δ_{TO} und Δ_i ab.

Oft wird angenommen, dass Δ_i von den Effizient-Anforderungen an den Fehlerdetektor abhängt, jedoch wird Δ_i durch das verwendete Netzwerk bestimmt: Die Detektionszeit für einen Fehler ist abhängig von Δ_i , der Übertragungszeit der Nachricht (Δ_t) und einer zusätzliche Spanne α zum Ausgleich von Verzögerungen ($\Delta_{TO} \approx \Delta_i + \alpha$). Ist Δ_i wesentlich kleiner als die Übertragungszeit, so hat eine weitere Verringerung keinen Einfluss auf die Detektionszeit, da diese nicht kürzer als die Übertragungszeit sein kann. Eine weitere Reduktion würde nur zu einer höheren Netzwerklast führen. Wählt man Δ_i wesentlich größer als die Übertragungszeit, wird Δ_i den größten Teil der Detektionszeit bestimmen. Somit ist es sinnvoll Δ_i nahe bei der durchschnittlichen Übertragungszeit zu wählen, außer die für den Fehlerdetektor bereitgestellte Netzwerklast wird dadurch überschritten. Somit wird der Parameter Δ_i nicht von den Qualitätsanforderungen, sondern von dem zugrundeliegendem Netzwerk bestimmt [11], [12].

Die Effizienz eines Fehlerdetektors hängt somit bei geeigneter Wahl von Δ_i nur noch von der Timeout-Zeit Δ_{TO} ab.

C. Adaptive Fehlerdetektoren

Adaptive Fehlerdetektoren haben das Ziel sich an Änderungen der Netzwerkbedingungen dynamisch anzupassen und damit bessere Abschätzungen für die Ankunftszeit des nächsten Heartbeats, also für Δ_{TO} , zu erhalten. Dafür existieren verschiedene Ansätze, die im Folgenden kurz vorgestellt werden:

a) Chen-Fehlerdetektor: Chens Fehlerdetektor verwendet einen Ansatz, basierend auf der probabilistische Analyse des Netzwerkverkehrs. Das Protokoll benutzt die Ankunftszeiten zuvor empfangener Heartbeats, um eine Abschätzungen für die Ankunftszeit des nächsten Heartbeats zu erhalten. Zu der errechneten Zeit wird ein konstanter Sicherheitsparameter,

der je nach Qualitätsanforderungen gewählt wird, addiert, um falsche Detektionen zu vermeiden. Die Ankunftszeit des nächsten Heartbeats wird nach jeder Ankunft eines Heartbeats erneut berechnet [21].

b) Bertier-Fehlerdetektor: Bertiers Fehlerdetektor beruht auf einem ähnlichen Ansatz wie Chens Fehlerdetektor. Zur Kalkulation der Ankunftzeiten wird Chens Abschätzung verwendet, jedoch wird der Sicherheitsparameter nicht konstant gewählt. Der Sicherheitsparameter wird nach Jacobsons Schätzung der round-trip time [8] gewählt. Dadurch ergeben sich kürzere Fehlerdetektionszeiten, jedoch sinkt auch die Genauigkeit. [10]

D. ACCRUAL Fehlerdetektoren

Accrual Fehlerdetektoren unterscheiden sich von herkömmliche Fehlerdetektoren dadurch, dass sie keine Liste von vermutlich abgestürzten Prozessen liefern. Stattdessen werden für jeden Prozess Werte einer kontinuierlichen Skala berechnet. Je höher dieser Wert, desto wahrscheinlicher ist es, dass ein Prozess abgestürzt ist.

Ein Accrual Fehlerdetektor bietet, im Vergleich zu einem herkömmlichen Fehlerdetekor, weniger Abstraktion und keine Interpretation der errechneten Werte. Der Vorteil davon liegt darin, dass eine Anwendung, die den Fehlerdetekor verwendet, selbst entscheiden kann, wann ein Prozess als abgestürzt betrachtet wird oder anhand der Werte andere angemessenen Verfahren ausführen kann. Ein Beispiel für einen Accrual Fehlerdetektor ist der φ -Fehlerdetektor [12].

III. ZUSTANDSSICHERUNG

Die Sicherung des Zustandes eines Dienstes oder einer Anwendung ist erforderlich, um bei einem Ausfall des Hauptsystems den Dienst auf einem Ersatzsystem in den Zustand vor dem Ausfall zurückversetzen zu können. Der Zustand wird meistens auf mehrere Ersatzsysteme repliziert, sodass bei einem Ausfall eines der Ersatzsysteme die Bearbeitung von Anfragen übernehmen kann. Die Herausforderungen dabei sind Transparenz und Konsistenz zwischen den Systemen mit möglichst wenig Overhead herzustellen [1].

Der Zustand eines Diensts ist aus drei Komponenten zusammengesetzt: der Identität des Dienstes, dem Verbindungszustand und dem Anwendungszustand.

Die Identität des Dienstes ist die Adresse, die von allen Clients verwendet wird, um einen Dienst in Anspruch zu nehmen. Im Falle eines Dienstes der auf dem *Transmission Control Protocol* [19] aufbaut, ist die Identität des Dienstes beispielsweise durch eine IP-Adresse und einem Port bestimmt [1].

Der Verbindungszustand eines Dienstes ist der Zustand, der notwendig ist, um die Kommunikation zu Clients aufrechtzuerhalten. Dies beinhaltet zum Beispiel empfangene und gesendete Nachrichten [1].

Der Anwendungszustand ist der interne Zustand der Anwendung, der Notwenig ist, um Anfragen von Clients korrekt zu beantworten. Weiterhin ist es möglich, dass ein Service zustandslos ist, also kein Anwendungzustand existiert, der gesichert werden muss [1].

Es existieren drei Haupttechniken, um den Zustand eines Dienstes oder einer Anwendung zu sichern: Replikation, Kontrollpunkte und Protokollierung.

A. Replikation

Zur Replikation des Zustands einer Anwendung existieren verschiedene Methoden. Ersatzsysteme, die mittels Replikation den Zustand einer Anwendung sichern, werden Replikanten genannt. Im Folgenden werden vier übliche Replikationsstrategien erläutert.

1) Aktive Replikation: Ziel der aktiven Replikation ist es, alle Replikanten permanent in einem gleichen Zustand zu halten, sodass ein Server leicht durch einen anderen ersetzt werden kann. Dazu werden alle Anfragen eines Clients an einen Dienst an alle Replikanten gesendet. Diese verarbeiten die Anfragen und senden alle dieselben Antworten an den Client. Die erste Antwort, die den Client erreicht, wird angenommen, die restlichen werden verworfen [16]. Abbildung 2 a) veranschaulicht die aktive Replikation.

Damit die Replikanten in denselben Zustand enden, müssen Anfragen in derselben Weise und in derselben Reihenfolge bearbeitet werden. Dazu kann ein atomares Multicastprotokoll verwendet werden. Dieses stellt sicher, dass entweder alle Replikanten eine Anfrage erhalten, oder keiner (Atomarität). Außerdem wird garantiert, dass alle Nachrichten in derselben Reihenfolge empfangen werden [16].

Damit alle Replikanten in dem selben Zustand enden, muss der Dienst deterministisch arbeiten. Das heißt, dass der Dienst bei gleichem Anfangszustand und gleicher Anfragesequenz, immer gleiche Antworten produziert und im selben Zustand endet. Viele Dienste und Anwendungen sind jedoch nicht deterministisch. Ein Grund hierfür ist zum Beispiel die Verwendung von Multithreading [16].

Sind die genannten Bedingungen erfüllt, können Verbindungen leicht und schnell zwischen den Replikanten ausgetauscht werden, da alle denselben Zustand und Informationen besitzen. Es werden jedoch viele Ressourcen benötigt, da alle Replikanten alle Anfragen bearbeiten müssen [1].

Bestehende Anwendungen können ebenfalls aktiv repliziert werden, ohne das eine Modifizierung des Client- oder des Servercodes notwendig ist. Dieses Vorgehen wird als transparente Replikation bezeichnet. Um transparente Replikation zu unterstützen, könnten alle ein- und ausgehenden Verbindungen durch ein atomares Multicast-Protokoll getunnelt werden. Dies ist möglich indem sowohl der Client-, als auch der Servercode mit einem Tunnelcode umhüllt wird (wrapping). Jedoch ist es in der Regel schwierig den Code von Clients zu modifizieren oder zu umhüllen, da kein Einfluss auf diese besteht. Eine Möglichkeit dies zu umgehen, ist die Verwendung eines serverseitigen Mullticastmechanismus. Die Modifizierung von Clients ist dabei nicht mehr notwendig, da nur die Server sich untereinander koordinieren, um die genannten Eigenschaften eines atomaren Multicasts sicherzustellen [7].

2) Semi-aktive Replikation: Bei Semi-aktiver Replikation handelt es sich um eine Erweiterung von aktiver Replikation, die die Anforderung des Determinismus eines Prozesses aufhebt. Dies ist durch die Verwendung eines leader/follower-Protokolls [15] möglich. Jede nicht deterministische Aktion

wird dabei zuerst von dem sogenannten Leader durchgeführt. Der Leader informiert alle übrigen Replikanten, die Follower, mit dem Ergebnis der erfolgreich durchgeführten Aktion. Durch diese Informationen können die Follower sich in einem zum Leader konsistenten Zustand halten [5].

3) Passive Replikation: Bei der Passive Replikation erhält nur ein Server, der Primary genannt wird, alle Anfragen an einen Dienst und verarbeitet diese. Daraufhin aktualisiert dieser den Zustand der anderen Replikanten und sendet, nachdem alle Replikanten den Erhalt des Zustandes bestätigt haben, die entsprechende Antwort an den Client [6]. Abbildung 2 b) illustriert die passive Replikation.

Durch dieses Vorgehen können viele Ressourcen gespart werden, da nicht alle Replikanten Anfragen bearbeiten müssen. Außerdem muss der Dienst nicht deterministisch arbeiten. Der Nachteil der passiven Replikation ist jedoch, dass die Antwortzeiten, im Vergleich zur aktiven Replikation, beim Ausfall des Primary erhöht sind. Dies liegt daran, dass ein Client eine Anfrage erneut versenden muss, wenn der Primary nach dem Erhalt der Anfrage ausfällt. Des Weiteren ist ein Mechanismus zur Wahl des Primary notwendig, wodurch die Antwortzeit im Falle eines Absturzes signifikant erhöht ist. Bei diesem Mechanismus handelt es sich üblicherweise um ein Gruppenmitgliedschaftsprotokoll. Dieses Protokoll fasst alle Ersatzsysteme in einer Gruppe zusammen und sorgt dafür, dass als ausgefallen angenommene Systeme aus dieser Gruppe entfernt werden [6].

4) Semi-passive Replikation: Semi-passive Replikation basiert auf den Charakteristiken von passiver Replikation. Es wird jedoch kein Gruppenmitgliedschaftsprotokoll zur Wahl eines *Primary* verwendet. Diese Problematik wird mittels eines "Lazy Consensus-Algorithmus" gelöst. Dadurch ist eine schnellere Reaktion auf Ausfälle möglich ist [6].

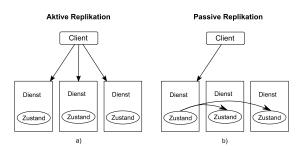


Abbildung 2. Aktive und passive Replikation

B. Kontrollpunkte

Mit Kontrollpunkten wird der Zustand einer Anwendung bei einem kritischen Zustandswechsel oder wenn eine bestimmte Zeit seit dem letzten Kontrollpunkt vergangen ist, in einem stabilen Speicher gesichert oder auf ein Ersatzsystem kopiert. Tritt ein Fehler auf, wird der Zustand des zuletzt gespeicherten Kontrollpunktes wiederhergestellt und die Verarbeitung fortgesetzt.

Kontrollpunkte für Anwendungen, die aus mehreren Prozessen bestehen, lassen sich in zwei Kategorien einteilen: consistent checkpointing und undependent checkpointing. Abbildung 3 zeigt beide Methoden im Vergleich.

- 1) consistent checkpointing: Beim consistent checkpointing synchronisieren sich alle Prozesse regelmäßig und speichern ihren aktuellen gesamten Zustand zur gleichen Zeit. Diese Methode erfordert einen konsistenten Zustand aller Prozesse, bei dem alle versendeten Nachrichten bei ihrem Ziel angekommen sein müssen. Jeder Prozess speichert seinen gesamten Zustand. Bei dem Auftreten eines Fehlers, wird jeder Prozess zurück in den gesicherten Zustand versetzt und die Ausführung fortgesetzt. Ein Vorteil diese Vorgehens ist, dass der Zustand für eine beliebige Anzahl an abgestürzten Prozessen wiederhergestellt werden kann. Außerdem kann der Overhead bei einem System mit wenig erwarteten Fehlern reduziert werden, indem die Zeit zwischen der Erstellung der Kontrollpunkte erhöht wird. Ein Nachteil der Methode ist jedoch, dass für die Erstellung eines Kontrollpunktes global kommuniziert werden muss. Des Weiteren werden bei einem Fehler alle Prozesse in einen zuvor gesicherten Zustand zurückversetzt, wodurch Daten verloren gehen können. [14].
- 2) independent checkpointing: Beim independent checkpointing wird eine globale Synchronisation und die Zurücksetzung aller Prozesse vermieden. Jeder Prozess sichert seinen Zustand unabhängig, immer wenn mit anderen Prozessen kommuniziert wird. Dieses Vorgehen ist sehr aufwendig, da bei jedem Zustandswechsel ein Kontrollpunkt erstellt werden muss. Aus diesem Grund werden in der Praxis hybride Techniken mit Protokollierung verwendet [14].

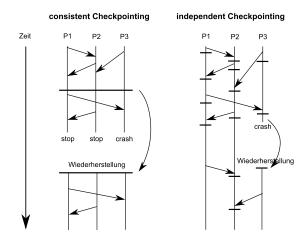


Abbildung 3. Consistent und independent Checkpointing [4]

C. Protokollierung

Ein weiterer Ansatz zur Sicherung des Anwendungszustands ist Protokollierung. Es werden alle eingehen Anfragen und bei nicht deterministischen Anwendungen auch alle erstellten Antworten redundant gesichert. Bei dem Ausfall eines Servers werden alle protokollierten Nachrichten zu einem alternativen Server wiedergegeben, sodass dieser zurück in den Ausgangszustand, vor dem Auftreten des Fehlers, versetzt wird. Je nach den Anforderungen kann auf verschiedenen Leveln protokolliert werden. Zum Beispiel können Nachrichten auf der Transportebene oder auf Anwendungsebene gesichert werden [2].

Bei einer Protokollierung ist der Overhead für die Verarbeitung ohne Fehler geringer, als zum Beispiel für aktive

Replikation. Jedoch sind die Zeiten für eine Zustandswiederherstellung erhöht, da alle Nachrichten wiedergegeben müssen, um den Zustand vor dem Fehler wiederherzustellen [1].

D. Hybride Verfahren

Da das häufige Sichern von Kontrollpunkten sehr aufwendig ist, existieren Techniken, die Kontrollpunkte und Protokollierung kombinieren. Nachrichten, die nach einem Kontrollpunkt empfangen werden, werden protokolliert. Nach einem Ausfall wird der zuletzt gesicherte Kontrollpunkt wiederhergestellt und die protokollierten Nachrichten wiedergegeben. Dadurch wird der Overhead im Vergleich zu einfachen Kontrollpunkten gesenkt, aber die Zeit, die für eine Wiederherstellung des Zustands benötigt wird, erhöht [1].

IV. FAILOVER UND SWITCHOVER

In diesem Abschnitt werden Techniken für den eigentlichen Fail-/Switchover, also den Wechsel von einem Hauptzu einem Ersatzsystem, beschrieben. Wird von einem Fehlerdetektor ein Ausfall des aktiven Systems erkannt, muss sichergestellt werden, dass der Dienst für bestehende und neue Verbindungen erhalten bleibt.

Es existieren viele Systeme, die während des Failovers nicht verfügbar sind, sodass neue Verbindungen nicht akzeptiert werden. Aktive Verbindungen werden so lange verzögert, bis ein stabiler, gültiger Zustand des Dienstes wiederhergestellt ist. Um einen Ausfall vor einem Client zu verstecken, sollte die Zeit für einen Failover möglichst kurz sein. Dies ist besonders bei Echtzeitanwendungen wie Multimediadiensten wichtig.

In diesem Abschnitt werden einige Möglichkeiten vorgestellt, die sicherstellen, dass Anfragen, im Falle eines Ausfalls, zu einem alternativen Server umgeleitet werden.

- 1) IP-Adressübernahme: Die einfachste Möglichkeit für einen Serverwechsel ist die Übernahme der IP-Adresse des ausgefallenen Servers. Dies ist zum Beispiel durch das Address Resolution Protocol (ARP) möglich. ARP ermittelt von IP-Adressen die entsprechenden physikalischen Hardwareadressen. Fällt ein Server aus, wird die IP-Adresse eines Ersatzsystems rekonfiguriert. Mittels ARP wird die Übernahme der IP-Adresse im Netzwerk bekannt gegeben, wodurch das Ersatzsystem die Position des ausgefallenen Servers einnimmt.
- 2) Domain Name System Failover: Es ist möglich das Domain Name System zu verwenden, um die Verfügbarkeit eines Systems zu erhöhen. Bevor ein Client eine Anfrage an einen bestimmten Hostnamen sendet, wird der Hostname zuerst mittels einer Anfrage an das DNS zu einer IP-Addresse aufgelöst. Die Verfügbarkeit eines Systems kann dadurch erhöht werden, dass abgestürzte Server erkannt und von DNS-Antworten ausgeschlossen werden. Es existieren weitere Techniken, die auf DNS basieren. Dazu gehören beispielsweise Round Robin DNS und DNS Aliasing Methoden [3].

Ein Failover mittels *DNS* erfordert, dass Clients wiederholt Anfragen senden, wenn sie keine Antwort erhalten. Durch DNS-Caching kann es dazu kommen, dass für eine gewisse Zeit alte, fehlerhafte Server zurückgegeben werden. Dadurch wird die Effizienz von *DNS*-Systemen gesenkt, da eine lange Zeit bis zur nächsten Aktualisierung des Caches vergehen kann [1].

3) Proxies: Eine Möglichkeit für einen Fail-/Switchover bieten Proxyserver. Clients kommunizieren nicht direkt mit einem Server, sondern verbinden sich mit einem Proxyserver, der die Schnittstelle zwischen Clients und verschiedenen Servern bereitstellt. Der Proxyserver nimmt Verbindungen von Clients entgegen und leitet die Anfragen an einen Server weiter. Wird mittels eines Fehlerdetektors der Ausfall des verwendeten Servers festgestellt oder wird ein Switchover initiiert, kann der Proxyserver weiterhin die Verbindung zum Client aufrecht erhalten. Ist der Anwendungszustand auf einem Ersatzsystem wiederhergestellt, kann der Proxy zu diesem eine neue Verbindung aufbauen und die Nutzung des Dienstes kann ununterbrochen und transparent zum Client fortgesetzt werden [17].

Ein großer Nachteil dieser Technik besteht darin, dass der Proxyserver einen Single Point of Failure darstellt. Bei einem Ausfall des Proxyservers kann der Dienst nicht weiter verwendet werden. Des Weiteren kann der Proxyserver auch einen Performance-Engpass darstellen [1].

4) Umleitung durch die Anwendungsschicht: Durch die Nutzung eines angemessenen Protokolls auf der Anwendungsebene ist es möglich Anfragen von Clients an aktive Server umzuleiten. Ein Beispiel hierfür bietet das Hypertext Transfer Protocol (HTTP) [13]. HTTP beinhaltet eine Funktion, die es ermöglicht Clients an andere Server weiterzuleiten, sodass diese eine Anfrage erneut an einen alternativen Server stellen. Diese Vorgehensweise kann erweitert werden, um einen Failoder Switchover zu ermöglichen. Ähnlich zu einem Proxy kann ein Server für die Umleitung verwendet werden. Dieser Server überprüft die Aktivität von Ersatzsystemen mittels eines Fehlerdetektors und leitet anfragende Clients an ein aktives System um. Die Clients bauen dann direkt zu einem Ersatzsystem eine Verbindung auf, sodass weiter Kommunikation nicht mehr über den Umleitungsserver laufen muss [1].

Der Nachteil dieser Vorgehensweise ist, dass der Umleitungsserver, wie ein Proxy, einen Single Point of Failure darstellt [1].

5) M-TCP: Bei M-TCP handelt es sich um ein verlässliches, verbindungsorientiertes Protokoll der Transportschicht, das die effiziente Migration von bestehenden Verbindungen unterstützt. Der Endpunkt einer aktiven Verbindung kann zu jeder Zeit, transparent für den Client, zwischen verschiedenen Servern gewechselt werden. Nach einer Verbindungsmigration ist eine nahtlose Fortsetzung der Nutzung eines Dienstes möglich.

M-TCP stellt diese Funktionalitäten unter bestimmten Anforderungen zur Verfügung, die meist als ein Vertrag zwischen dem Transportprotokoll und der nutzenden Anwendung beschrieben werden. Die Anwendung verpflichtet sich Funktionalitäten zur Exportierung und Importierung des Anwendungszustands für eine bestimmte Verbindung bereitzustellen. Im Gegenzug dazu überträgt das Protokoll diesen Zustand zu einem neuem Server und synchronisiert den Zustand der Anwendung mit dem des Transportprotokolls.

Abbildung 4 zeigt ein Beispiel für eine Verbindungsmigration. Ein Client kontaktiert einen Server S1 durch eine Verbindung C_{id} . Beim Verbindungsaufbau stellt der Server die Adresse eines alternativen Servers und ein Migrierungs-

zertifikat bereit. M-TCP initiiert einen Verbindungswechsel zu Server S2, indem eine Verbindung aufgebaut und das Migrierungszertifikat übertragen wird. Daraufhin überträgt M-TCP den Anwendungs- und Verbindungszustand zum alternativen Server, wo die Verbindung reibungslos fortgesetzt wird [18], [20].

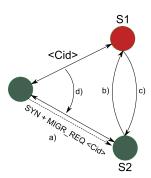


Abbildung 4. Migrationsmechanismus in M-TCP [20]

6) Stream Control Transmission Protocol: Das Stream Control Transmission Protocol (SCTP) zeichnet sich durch die Möglichkeit für Multihoming aus. Dadurch kann eine einzelne Verbindung mit verschiedenen IP-Adressen assoziieren werden. Während einer normalen Verbindung wird ein einzelner Endpunkt ausgewählt, an den alle Anfragen gesendet werden. Ist dieser Endpunkt nicht mehr erreichbar, erkennt SCTP diesen Fehler und führt einen temporären Failover zu einer angegebenen alternativen Adresse durch. Hierfür ist keinerlei Interaktion des Nutzers oder der Anwendung notwendig. Während Anfragen an den neuen Server gesendet werden, wird mittels Heartbeats überprüft, ob der ursprüngliche Server wieder erreichbar ist. Wenn dies der Fall ist, wird der Failover rückgängig gemacht und der ursprüngliche Server verwendet.

Die Zeit für die Fehlererkennung für den empfohlenen Standard beträgt mindestens 63 und maximal 360 Sekunden. Dies ist für viele Anwendungen unzureichend. Um diese Zeit zu reduzieren können jedoch spezielle Parameter angepasst werden [9].

V. Fazit

In dieser Arbeit wurde eine Übersicht über die Hauptkomponenten eines Fail-/Switchover-Systems geliefert. Zu Beginn wurden verschiedene Ansätze für Fehlerdetektoren vorgestellt. Daraufhin wurden Techniken zur Zustandssicherung eines Dienstes oder einer Anwendung betrachtet. Zum Schluss wurden Möglichkeiten für einen Serverwechsel erörtert. Insgesamt besitzen alle vorgestellten Techniken und Ansätze spezielle Vor- und Nachteile. Der Einsatz einer jeweiligen Komponente muss je nach Anwendungszweck und den Anforderungen an das System untersucht werden. Stehen zum Beispiel viele Ressourcen zur Verfügung und ist ein schneller Fail- oder Switchover für die bereitgestellte Anwendung vonnöten, sollte aktive Replikation verwendet werden. Bei weniger zur Verfügung stehenden Ressourcen können durch passive Replikation und der Verwendung eines speziellen Transportprotokolls wie dem modifizierten SCTP dennoch relativ gute Fail-/Switchover-Zeiten erreicht werden. Insgesamt muss immer zwischen den entstehenden Kosten für einen Ansatz und den notwendigen Anforderungen an den Fail- oder Switchover abgewägt werden.

LITERATUR

- N. Aghdaie. Transparent fault-tolerant network services using off-theshelf components. UCLA, 2005. III, III-A1, III-C, III-D, IV-2, IV-3, IV-4
- [2] Lorenzo Alvisi, Thomas C. Bressoud, Ayman El-Khashab, Keith Marzullo, and Dmitrii Zagorodnov. Wrapping server-side tcp to mask connection failures. In *In Proc. IEEE INFOCOM*, pages 329–337, 2001.
 III-C
- [3] T. Brisco. Dns support for load balancing, 1995. IV-2
- [4] Monica S. Lam Daniel J. Scales. Transparent fault tolerance for parallel applications on networks of workstations, 1996. 3
- [5] Xavier Defago and Andre Schiper. Specification of replication techniques, semi-passive replication, and lazy consensus. Technical report, Japan Advanced Institute of Science and Technology, 2002. III-A2
- [6] Xavier Defago, André Schiper, and Nicole Sergent. Semi-passive replication. In IN PROCEEDINGS OF THE 17TH IEEE SYMPOSIUM ON RELIABLE DISTRIBUTED SYSTEMS (SRDS, pages 43–50, 1998. III-A3, III-A4
- [7] Christof Fetzer and Neeraj Suri. Practical aspects of ip take-over mechanisms. In In Proceedings of 9 th IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS 2003), Capri Island, 2003. III-A1
- [8] V. Jacobson. Congestion avoidance and control. In Symposium proceedings on Communications architectures and protocols, SIGCOMM '88, pages 314–329, New York, NY, USA, 1988. ACM. II-C0b
- [9] Armando L. Caro Jr., Paul D. Amer, and Randall R. Stewart. Rethinking end-to-end failover with transport layer multihoming. *Annales des Télécommunications*, 61(1-2):92–114, 2006. IV-6
- [10] Pierre SENS Marin BERTIER, Olivier MARIN. Implementation and performance evaluation of an adaptable failure detector, 2002. II-B1, II-B2, II-B3, II-C0b
- [11] Maxime Müller. Performance evaluation of a failure detector using snmp, 2003. II-B3
- [12] Rami Yared Takuya Katayama Naohiro Hayashibara, Xavier Defago. The phi accrual failure detector, 2004. II-B3, II-D
- [13] J. Mogul H. Frystyk L. Masinter P. Leach T. Berners-Lee R. Fielding, J. Gettys. Hypertext transfer protocol – http/1.1, 1999. IV-4
- [14] Daniel J. Scales and Monica S. Lam. Transparent fault tolerance for parallel applications on networks of workstations. In *In Proceedings of the USENIX 1996 Annual Winter Technical Conference*, 1996. III-B1, III-B2
- [15] Douglas C. Schmidt, Carlos O'Ryan, Michael Kircher, Irfan Pyarali, and Back Back-endend. Leader/followers a design pattern for efficient multi-threaded event demultiplexing and dispatching. In *University of Washington. http://www.cs.wustl.edu/ schmidt/PDF/lf.pdf*, pages 00–29. AddisonWesley, 2000. III-A2
- [16] Fred B. Schneider and Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. ACM Computing Surveys, 22:299–319, 1990. III-A1
- [17] Oliver Spatscheck, Jørgen S. Hansen, John H. Hartman, and Larry L. Peterson. Optimizing tcp forwarder performance. *IEEE/ACM Transactions on Networking*, 8:146–157, 1998. IV-3
- [18] Kiran Srinivasan. Mtcp: Transport layer support for highly available network services, 2001. IV-5
- [19] W. Richard Stevens. TCP/IP illustrated (vol. 1): the protocols. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1993. III
- [20] Florin Sultan, Kiran Srinivasan, Deepa Iyer, and Liviu Iftode Ý. Migratory tcp: Connection migration for service continuity in the internet. In In Proc. ICDCS 2002, pages 469–470, 2002. IV-5, 4
- [21] Marcos Kawazoe Aguilera Wei Chen, Sam Toueg. On the quality of service of failure detectors, 2002. II-A, II-B1, II-C0a

Bestimmung des Anonymitätsgrads

Nadine Trüschler

Zusammenfassung—Personenbezogene Daten zu veröffentlichen, ohne sensitive Informationen zu enthüllen, und die Anonymisierung von Kommunikationen stellen noch immer essenzielle Probleme dar. Verschiedene Metriken wurden entwickelt, um den Grad der Anonymität eines Anonymisierungsverfahrens zu bestimmen. Gewisse Metriken berücksichtigen manche Zusammenhänge nicht und geben daher ein unzureichendes Verständnis an Anonymität. Solche Schwachstellen wurden aufgedeckt, die Anlass zur Entwicklung besserer Metriken geben. Erste Ansätze sind bereits zu erkennen. So bauen einige Metriken auf anderen auf, beispielsweise t-Closeness auf l-Diversity. Es bedarf kontinuierlicher Forschung, um weitere nicht betrachtete Zusammenhänge ermitteln und überarbeiten zu können. In diesem Paper werden die wichtigsten Metriken zusammengefasst und auf Zusammenhänge und Gemeinsamkeiten untersucht.

I. Einführung

Anonymität ist nicht nur im Internet vertreten. Sie ist auch in gewissem Maße auf der Straße oder beim Einkaufen gegenwärtig. Betrachtet man das Beispiel in einem Supermarkt, muss man sich lediglich in bestimmten Situationen ausweisen, beispielsweise beim Kauf von Alkohol. Hier ist demnach deutlich zu erkennen, wann man anonym ist und wann nicht, denn der Kunde selbst veranlasst einen Grund für seine Identifizierung. Doch auch ohne sich auszuweisen, ist man nicht immer anonym. Befindet man sich beispielsweise in einem kleinen Geschäft, in dem sich nur sehr wenige Kunden aufhalten, ist Anonymität nicht zwingend gegeben.

Wenn man in die digitale Welt des Internets wechselt, verändert sich die Situation. Allein bei dem Besuch einer Webseite wird die IP Adresse übermittelt. Selbst dynamische IP Adressen geben Aufschluss über die Verbindung zwischen dem Benutzer und der IP Adresse. Sogar wenn Benutzern diese Verbindung bekannt ist, fühlen sie sich auch ohne Einsatz von Anonymisierung in der digitalen Welt anonymer. Böswillige Angreifer könnten mangelnde Anonymität ausnutzen. Daher zeigen viele Unternehmen Interesse am Beschaffen und Ansammeln von Benutzerprofilen. Mit diesen Daten sollen aktuelle und zukünftige Netzwerk- und Datensicherheitsprobleme gelöst und die Privatsphäre geschützt werden. Um mit diesen Daten arbeiten zu können, müssen sie anonymisiert werden. Daher sind Unternehmen immer mehr daran interessiert, die modernsten Fortschritte der Anonymisierungstechniken anzuwenden. Auch die Anonymisierung von Kommunikationen ist für Unternehmen von Bedeutung. Solche Anonymisierungstechniken können weiter unterteilt werden. So wurden Datenmetriken entwickelt, die sich ausschließlich damit befassen, den Anonymitätsgrad des Anonynimiserens von Benutzerdaten zu bewerten. Erarbeitete Netzwerkmetriken hingegen bewerten den Anonymitätsgrad der Verschleierung von Kommunikation. In diesem Paper werden einige Metriken dargetsellt, die den Grad an Anonymität eines Anonymisierungsverfahrens bestimmen, und auf Übereinstimmungen und Gemeinsamkeiten

untersucht.

Das Paper beginnt mit einigen Begriffsdefinitionen, die zum Verständnis der weiteren Kapitel nötig sind. Kapitel III definiert die wichtigsten Metriken zur Anonymisierung von Datensätzen. Kapitel IV befasst sich mit Anonymitätsmetriken für Kommunikationsnetze. Die wichtigsten Metriken werden in Kapitel V zusammengefasst und auf Zusammenhänge und Gemeinsamkeiten untersucht. Das Paper schließt mit einer Konklusion.

II. BEGRIFFSDEFINITIONEN

Zu Beginn werden einige grundlegende Begriffe unter anderem von Pfitzmann und Köhntopp [7] definiert, die zum Verständnis der nachfolgenden Kapitel nötig sind.

A. Anonymität

In einer Menge an Subjekten N (z.B. Personen), der Anonymitätsmenge, ist Anonymität der Zustand, nicht identifizierbar zu sein. Dabei bedeutet "nicht identifizierbar", dass mit einer Wahrscheinlichkeit von $\frac{(N-1)}{N}$ nicht zugeordnet werden kann, welches Subjekt zugehörig ist.

B. Unbeobachtbarkeit

Unbeobachtbarkeit ist der Zustand einer Menge, in der ein Element des Interesses ununterscheidbar zu jedem anderen Element des Interesses ist. Dabei bedeutet ununterscheidbar, dass mit einer Wahrscheinlichkeit von $\frac{(l-1)}{l}$, wobei $l \leq N$ die Anzahl der Elemente des Interesses ist, keine Unterscheidung zwischen den Elementen möglich ist.

C. Pseudonymität

Pseudonymität ist der Gebrauch von Pseudonymen als IDs. Dabei ist ein Identifizierer (ID) eine zugewiesene Bezeichnung zur eindeutigen Identifizierung eines Objektes.

D. Größe der Anonymitätsmenge

Kelly et al. [4] beschreiben, dass sich die Größe der Anonymitätsmenge, oder bei Datenschutz die Größe der Menge an Äquivalenzklassen, für anomysierte Tabellen oder anonyme Netzwerke eignet. Kennt beispielsweise ein Angreifer die Anzahl an Benutzern N und kann C Benutzer während seines Angriffs ausschließen, so beträgt die Größe der Anonymitätsmenge n=N-C. Diese Größe bestimmt den Grad an Anonymität, der erreicht wurde. N kann sich auf mögliche Sender, Empfänger oder beide Benutzer einer Kommunikation beziehen. Dieses Beispiel lässt sich auch auf die Datenanonymisierung übertragen.

E. Explizite Identifizierer

Explizite Identifizierer sind Attribute, wie etwa eine Sozialversicherungsnummer, die eindeutig Personen identifizieren.

F. Quasi-Identifizierer

Quasi-Identifizierer sind Attribute, beispielsweise Geschlecht, deren Werte in Verbindung mit anderen eine Person identifizieren können.

G. Sensitive Attribute

Sensitive Attribute sind vertrauliche Daten wie beispielsweise Krankheit oder Gehalt, die es zu schützen gilt.

H. k-Anonymität

Für k-Anonymität, die von Sweeney [11] eingeführt wurde, so erklären Kelly et al. [4], müssen für einen Angreifer mindestens k Benutzer oder Benutzerpaare in der Anonymitätsmenge enthalten sein. In einer Datentabelle bedeutet k-Anonymität, dass für jedes Tupel mindestens k-1 weitere ununterscheidbare Tupel bezüglich der Menge an Quasi-Identifizierern vorhanden sind. Daher müssen explizite und Quasi-Identifizierer (z.B. Geburtstag oder Geschlecht) so minimal wie möglich generalisiert und/oder unterdrückt werden, sodass die Daten der Personen anonymisiert werden.

I. Individueller Anonymitätsgrad

Kelly et al. [4] führen die Skalierung in Abbildung 1 als individuellen Anonymitätsgrad ein.

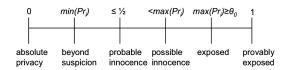


Abbildung 1. Skalierung des individuellen Anonymitätsgrads. [4]

Absolute privacy besagt, dass ein Benutzer i nie eine Nachricht sendet oder nicht in der Anonymitätsmenge AS enthalten ist, sodass die Wahrscheinlich gleich null ist $(Pr_i=0)$. Beyond suspicion bedeutet, es ist nicht wahrscheinlicher, dass ein Benutzer i die Nachricht gesendet hat, als jemand anderes. Probable innocence meint, es ist nicht wahrscheinlicher, dass ein Benutzer i die Nachricht gesendet hat, als sie nicht gesendet zu haben. Possible innocence bedeutet, dass eine nicht triviale Wahrscheinlichkeit besteht, ein anderer Benutzer als i sendete die Nachricht. Exposed hingegen bedeutet, dass ein Benutzer i mit signifikanter Wahrscheinlichkeit die Nachtricht gesendet hat, also $Pr_i = \max(Pr_j) \geq \theta_0, \forall j \in AS$. Bei provably exposed ist keine Anonymität mehr vorhanden, $Pr_i = 1$ und $Pr_j = 0, \forall j \in AS, i \neq j$. Dabei ist θ_0 eine Grenzwahrscheinlichkeit.

III. METRIKEN ZUR DATENSATZANONYMITÄT

Für Metriken, die die Anonymität von Datensätzen messen, sind als Ausgangspunkt Daten zu betrachten, die von Organisationen beispielsweise für Forschungszwecke veröffentlicht werden. Meistens handelt es sich hierbei um medizinische oder Befragungsdaten. Diese Daten werden in Tabellen gespeichert, von denen sich jede Reihe auf eine Person bezieht. Die Datensätze lassen sich in drei Kategorien aufteilen: (1) Explizite Identifizierer, (2) Quasi-Identifizierer und (3) sensitive Attribute. Um Daten zu anonymisieren, wird oft generalisiert. Dabei werden die Werte der Ouasi-Identifizierer mit Werten, die genereller sind, ersetzt. Resultierend bestehen mehr Datensätze aus der gleichen Menge an Werten der Quasi-Identifizierer. In diesem Kapitel werden die Metriken l-Diversity und t-Closeness, p-sensitive k-Anonymität und L1 Gleichartigkeit betrachtet, die sich gezielt mit der Anonymisierung von Datensätzen befassen.

A. l-Diversity und t-Closeness

Li et al. [5] entwickelten einen neuen Privatsphärenbegriff, t-closeness, da l-diversity in einigen Punkten keinen Schutz vor Attributenthüllung bieten kann.

Li et al. [5] definieren eine Äquivalenzklasse einer anonymisierten Tabelle als eine Menge an Datensätzen, die als Quasi-Identifizierer die gleichen Werte besitzen. Samarati und Sweeney [8] [9] führten k-anonymity ein, um Enthüllung einzugrenzen. So fordert k-anonymity, dass in jeder Äquivalenzklasse mindestens k Datensätze enthalten sind. Aber Li et al. zeigen, dass k-anonymity keine Attributenthüllung vermeiden kann. Dafür wurde von Machanavajjhala et al. [6] die sogenannte l-diversity definiert, welche verlangt, dass die Verteilung eines sensitiven Attributs in jeder Äquivalenzklasse mindestens l "gut-verteilte" Werte besitzt. Doch auch zu dieser Metrik lassen sich Probleme finden, die Li et al. einen Grund geben, um die neue Metrik t-closeness zu entwickeln. Diese fordert einen geringen Abstand zwischen der Verteilung eines sensitiven Attributs in jeder Äquivalenzklasse und der Verteilung eines Attributs in der Gesamttabelle. Betrachte man das folgende Beispiel: Zu Beginn hat ein Beobachter eine vorausgehende Meinung B_0 über das sensitive Attribut einer Person. Der Beobachter vermutet also das sensitive Attribut einer Person in der Tabelle zuordnen zu können. Danach erhält er eine vollkommen generalisierte Version der Datensätze, in der alle Attribute der Quasi-Identifizierer entfernt oder generalisiert wurden. Die Verteilung des sensitiven Attributwerts in der Gesamttabelle Q beeinflusst die Meinung des Beobachters. So entwickelt sich B_1 . Schließlich erhält der Beobachter die veröffentliche Tabelle. Da ihm die Werte der Quasi-Identifizierer der Person bekannt sind, kann er die Äquivalenzklassen, in denen sich die Datensätze der Person befinden, identifizieren und die Verteilung P an sensitiven Attributwerten in dieser Klasse lernen. So ändert sich seine Meinung zu B_2 . Im Vergleich zur l-diversity, die den Unterschied zwischen B_0 und B_2 verkleinert, bemüht sich tcloseness um die Verringerung des Unterschieds zwischen B_1 und B_2 . Li et al. nehmen also an, dass \mathbf{Q} , die Verteilung des sensitiven Attributs in der Gesamtpopulation der Tabelle, eine öffentliche Information ist. So wird ausschließlich das Maß, zu dem der Beobachter zusätzliche Informationen über spezielle Personen lernen kann, verringert. Nun gilt es, die Distanz zwischen zwei Wahrscheinlichkeitsverteilungen zu messen. Hierzu bedienen sich Li et al. der Earth Mover's Distanz (EMD). Sei $\mathbf{P}=(p_1,p_2,...,p_m)$, $\mathbf{Q}=(q_1,q_2,...,q_m)$ und d_{ij} die Grunddistanz zwischen dem Element i von \mathbf{P} und dem Element j von \mathbf{Q} . Es gilt einen Ablauf $F=[f_{ij}]$ zu finden, bei dem f_{ij} der Übergang der Masse des Elements i von \mathbf{P} zum Element j von \mathbf{Q} ist, welcher die Gesamtarbeit minimiert:

$$WORK(\mathbf{P}, \mathbf{Q}, F) = \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} f_{ij}$$
 (1)

wobei folgende Beschränkungen gelten:

$$f_{ij} \ge 0 \quad 1 \le i \le m, 1 \le j \le m \tag{2}$$

$$p_i - \sum_{j=1}^m f_{ij} + \sum_{j=1}^m f_{ij} = q_i \quad 1 \le i \le m$$
 (3)

$$\sum_{i=1}^{m} \sum_{j=1}^{m} f_{ij} = \sum_{i=1}^{m} p_i = \sum_{i=1}^{m} q_i = 1$$
 (4)

Aufgrund dieser drei Beschränkungen wird \mathbf{P} durch den Massenablauf F zu \mathbf{Q} umgewandelt. Die EMD ist als die Gesamtarbeit zu definieren, beispielsweise:

$$D[\mathbf{P}, \mathbf{Q}] = WORK(\mathbf{P}, \mathbf{Q}, F) = \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} f_{ij}$$
 (5)

So ist Anonymität gegeben, wenn $0 \le D[\mathbf{P}, \mathbf{Q}] \le t$.

B. p-sensitive k-Anonymität

Auch die Metrik *p*-sensitive *k*-Anonymität von Truta et al. [12] nimmt *k*-Anonymität zur Grundlage und gibt Aufschluss über das erreichte Maß an Anonymität. Ein veröffentlichter Datensatz erfüllt *p*-sensitive *k*-Anonymität, wenn er *k*-anonym ist und für jede Tupelgruppe an identischen Kombinationen von Hauptattributwerten des Datensatzes die Anzahl an verschiedenen Werten für jedes vertrauenswürdige Attribut mindestens *p* mal in der selben Gruppe vorkommt. Ein Datensatz muss zwei Bedingungen erfüllen:

- 1) Der veröffentlichte Datensatz muss die Eigenschaft von k-Anonymität mit $k \ge 2$ und
- 2) die Eigenschaft von p-sensitive k-Anonymität mit $p \ge 2$ erfüllen.

Um zu prüfen, ob der überarbeitete Datensatz die Bedingungen für p-sensitive k-Anonymität erfüllt, entwickelten Truta et al. einen Algorithmus (Algorithm 2 in [12]), der zwei Konditionen verwendet und sich so vom Basis Algorithmus (Algorithm 1 in [12]) unterscheidet und die Laufzeit verbessert. Dabei ist M ein Datensatz, n die Anzahl an Tupeln in M und q die Anzahl an vertrauenswürdigen Attributen in M. s_j stellt die Anzahl an verschiedenen Werten für das Attribut S_j , $(l \leq j \leq q)$ dar und f_i^j ein absteigend sortiertes Frequency Set der vertrauenswürdigen Attribute S_j , $(l \leq j \leq q)$ und $1 \leq i \leq s_j$. cf_i^j ist das gesondert betrachtete kumulativ absteigend sortierte Frequency Set der vertrauenswürdigen

Attribute S_j , $(l \leq j \leq q \text{ und } 1 \leq i \leq s_j)$. Des Weiteren ist cf_i wie folgt definiert: $cf_i = \max_{j=1,q} (cf_i^j) (1 \leq i \leq \min_{j=1,q} (s_j))$. Wobei ein Frequency Set eine Abbildung von jeder einzelnen Kombination an Werten von einer Menge SA zur Gesamtanzahl an Tupeln in einem Datensatz M mit den Werten von SA ist. Kondition 1 besagt, dass die kleinste Anzahl an verschiedenen Werten für vertrauenswürdige Attribute größer oder gleich p sein muss. Kondition 2 schränkt weiter ein, indem sie festlegt, dass die maximal erlaubte Anzahl an Kombinationen der Hauptattributwerte in dem veröffentlichten Datensatz

 $\min_{i=1,p-1} \left\lceil \frac{n - cf_{p-1}}{i} \right\rceil \tag{6}$

ist.

C. L1 Gleichartigkeit

Kelly et al. [4] beschreiben die L1 Gleichartigkeit als informationstheoretische Metrik sim(X,Y).

$$sim(X,Y) = 2 - \sum_{z \in X \cup Y} |P(X=z) - P(Y=z)|$$
 (7)

Sie berechnet die Differenz zwischen einem anonimysierten Objekt X und einem nicht anonymisierten Objekt Y, die beide extrahierbare Verteilungscharakteristiken besitzen. Anonymität ist dabei gewährt, wenn sim(X,Y)=2.

IV. METRIKEN ZUR KOMMUNIKATIONSANONYMITÄT

Metriken der Netzwerkanonymität bemessen die Anonymität von kommunizierenden Benutzern oder Verbindungen. Als Ausgangspunkt ist ein Anonymitätssystem, beispielsweise ein Mix Netzwerk zu betrachten. In solch einem Anonymitätssystem ist die Kommunikation zwischen Sender und Empfänger anonym.

Dazu werden das kombinatorische Anonymitätsmaß, eine informationstheoretische Metrik basierend auf Anonymitätswahrscheinlichkeitsverteilungen, zonenbasierte Empfänger k-Anonymität, Nachweistheorienanonymität und die Vereinigung von Netzwerk- und Anwendungsschicht betrachtet.

A. Kombinatorisches Anonymitätsmaß

Edman et al. [2] definieren eine Metrik, die auf der Permanenten einer Matrix basiert. Anhand dieser Metrik kann die Menge an Informationen ermittelt werden, die notwendig ist, um Sender bzw. Empfänger einer Nachricht oder Kommunikationspartner zu enthüllen. Ausgegangen wird von einem Anonymitätssystem wie beispielsweise einem Mix Netzwerk. In solch einem System ist die Kommunikation zwischen Sender und Empfänger anonym. Dazu wird die Menge $S = s_i$ als Eingabe und die Menge $T = t_i$ als Ausgabe definiert. Dabei ist die Eingabe gleich der Menge an Sendern und die Ausgabe gleich der Menge an Empfängern. Diesbezüglich stellen Edman et al. einen Bipartiten Graphen $G = (V_1, V_2, E)$ auf, der das System repräsentiert. Hierbei gilt $V_1 = S, V_2 = T$ und E stellt die Menge an Kanten dar, die für alle möglichen Abbildungen (s_i, t_i) steht. Anhand der Verzögerung Δ_i , die eine Nachricht m benötigt, um ein Anonymitätssystem zu durchlaufen, kann die Menge an möglichen Abbildungen zwischen Einund Ausgabe eingeschränkt werden. Diese Verzögerung liegt zwischen einem gegebenen Δ_{min} und Δ_{max} , also $\Delta_{min} \leq \Delta_i \leq \Delta_{max}$. Der Bipartite Graph G kann anhand der Eintritts- und Austrittszeitpunkte der Nachrichten, die die Knoten V_1 und V_2 bilden, aufgestellt werden. Für jedes s_i und t_j für die $\Delta_{min} \leq (t_j - s_i) \leq \Delta_{max}$ gilt, ist eine Kante in G zu finden, die die Knoten von s_i und t_j verbindet.

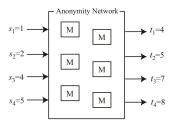


Abbildung 2. Mix Netzwerk mit vier Nachrichten. [2]

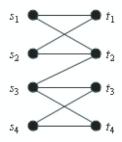


Abbildung 3. Bipartiter Graph für $\Delta_{min}=1$ und $\Delta_{max}=4$. [2]

Abbildung 2 zeigt beispielhaft ein Mix Netzwerk, welches von vier Nachrichten zu bestimmten Eintritts- und Austrittszeitpunkte durchlaufen wird. Nimmt man $\Delta_{min}=1$ und $\Delta_{max}=4$ an, wird die Nachricht, die zum Zeitpunkt $s_1=1$ das System betritt, das System im Intervall [2, 5] verlassen. Aus Abbildung 2 resultieren daher zwei Kanten im Bipartiten Graphen G, nämlich (s_1,t_1) und (s_1,t_2) . Verfährt man so mit den restlichen Eingaben s_2,s_3 und s_4 , erhält man den Bipartiten Graphen aus Abbildung 3.

Ein Bipartiter Graph kann auch als Adjazenzmatrix A dargestellt werden. So wird A(u,v)=1 gesetzt, wenn die Kante (u,v) in G existiert, wobei $u\in V_1$ und $v\in V_2$. Andernfalls wird A(u,v) auf 0 gesetzt. Die Permanente von A repräsentiert die Anzahl an perfekten Abbildungen in G.

$$per(A) = \sum_{\pi} \prod_{i=1}^{n} A(i, \pi(i))$$
 (8)

Um das Anonymitätslevel eines Systems zu bestimmen, wird eine $n \times n(0,1)$ -Matrix A benötigt, die alle möglichen Eingabe-Ausgabe-Beziehungen vertritt. Dieses Level an Anonymität wird wie folgt definietert:

$$d(A) = \begin{cases} 0 & \text{wenn } n = 1\\ \frac{\log(per(A))}{\log(n!)} & \text{wenn } n > 1 \end{cases}$$
(9)

wobei n! die Permanente der vollbesetzten (0,1)-Matrix J der Größe $n \times n$ ist. Die Werte liegen im Bereich [0,1]. Dabei bedeutet ein Wert von 0, dass keine Anonymität vorliegt, und ein Wert von 1 maximale Anonymität erschließen lässt.

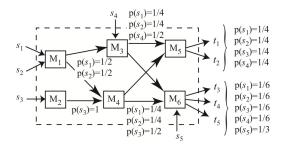


Abbildung 4. Mix Netzwerk mit fünf Nachrichten. [2]

Mit Hilfe von einer doppelt-stochastischen Matrix generalisieren Edman et al. ihre Metrik, wahrscheinlichkeitstheoretische Informationen mit einzubeziehen. Nimmt man das Mix Netzwerk aus Abbildung 4 zur Grundlage, lässt sich daraus eine doppelt-stochastische Matrix aufstellen. Um die Wahrscheinlichkeitsverteilung der ausgehenden Verbindungen der einzelnen Mixe zu berechnen, werden die Verteilungen der selben eingehenden Verbindungen des Mixes addiert und letztendlich mit der Anzahl an ausgehenden Verbindungen dividiert. So erhält man folgende doppelt-stochastische Matrix:

Der Grad an Anonymität lässt sich dann wie folgt bestimmen:

$$d(P) = \begin{cases} 0 & \text{wenn } n = 1\\ \frac{\log(per(P))}{\log(\frac{n!}{n^n})} & \text{wenn } n > 1 \end{cases}$$
 (11)

wobei $\frac{n!}{n^n}$ der kleinste Wert der Permanente einer $n \times n$ doppelt-stochastischen Matrix ist.

Gierlichs et al. [3] zeigen, dass die Metrik von Edman et al. [2] für das Senden oder Empfangen von mehreren Nachrichten ungeeignet ist. Dies lässt sich am besten anhand eines Beispiels verdeutlichen (Abbildung 5). Da in beiden Runden n! = 6 perfekte Abbildungen bestehen, ist das Anonymitätslevel des Systems maximal:

$$d(A) = \frac{\log(per(A))}{\log(n!)} = \frac{\log(n!)}{\log(n!)} = 1.$$
 (12)

Betrachtet man Runde 1, so lassen sich folgende sechs perfekte Abbildungen entnehmen:

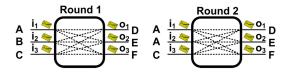


Abbildung 5. Zwei Runden eines Mix Netzwerks. [3]

{AE, AD, CF}, {AF, AD, CE}, {AF, AE, CD}. Diese lassen sich noch vereinfachen zu: {AD, AE, CF}, {AD, AF, CE}, {AE, AF, CD}.

Daher ist in diesem Fall keine perfekte Anonymität gegeben. Die Vielzahl an Sendern <u>oder</u> Empfängern wird von Gierlichs et al. durch Revidierung der Metrik von Edman et al. miteinbezogen. Notwendig ist hierfür die Ermittlung der Anzahl an Äquivalenzklassen im Fall von mehreren Sendern und im Fall von mehreren Empfängern. Für eine Multimenge an Sendern S^* sendet jeder Sender s_j eine Vielzahl n_j an Nachrichten. Allerdings müssen dabei die Empfänger eine Menge und keine Multimenge bilden (z.B. $R^* = R$). So ist die Größe jeder Äquivalenzklasse $\prod_{j=1}^{\sigma} n_j!$ und die Anzahl an Äquivalenzklassen:

$$\Xi = \frac{per(A)}{\prod_{i=1}^{\sigma} n_i!}.$$
(13)

Für eine Multimenge an Empfängern R^* empfängt jeder Empfänger r_l eine Vielzahl k_l an Nachrichten. Allerdings müssen dabei die Sender eine Menge und keine Multimenge bilden (z.B. $S^*=S$). So ist die Größe jeder Äquivalenzklasse $\prod_{l=1}^{\rho} k_l!$ und die Anzahl an Äquivalenzklassen:

$$\Psi = \frac{per(A)}{\prod_{l=1}^{\rho} k_l!}.$$
(14)

So wird das Anonymitätslevel des System neu definiert als:

$$d^*(A) = \begin{cases} 0 & \text{wenn } n = 1 \\ \frac{\log \Xi}{\log(n!)} & \text{wenn } R^* = R \text{ und } n > 1 \\ \frac{\log \Psi}{\log(n!)} & \text{wenn } S^* = S \text{ und } n > 1 \\ \frac{\log(per(A))}{\log(n!)} & \text{wenn } R^* = R \text{ und } S^* = S \text{ und } n > 1. \end{cases}$$

$$(15)$$

Doch auch der Fall von Multimengen an Sender und Empfänger wird von Gierlichs et al. genauer betrachtet. M ist die Menge, die alle möglichen perfekten Abbildungen im Graphen G beinaltet. $[M_P]$ stellt die Äquivalenzklassen eines Elements $M_P \in M$ dar und ist die Teilmenge aller Elemente in M, die zu M_P äquivalent sind. Θ ist die Anzahl der Äquivalenzklassen und C_P die Anzahl an äquivalenten perfekten Abbildungen in der Klasse $[M_P]$. Zur Berechnung des Anonymitätslevels des Systems muss miteinbezogen werden, mit welcher Wahrscheinlichkeit eine korrekte perfekte Abbildung M_C einer Äquivalenzklasse $[M_P]$ zugehört.

$$Pr(M_C \in [M_P]) = \frac{C_P}{per(A)} \tag{16}$$

So wird das Anonymitätslevel des Systems schließlich wie

folgt generalisiert:

$$d^*(A) = \begin{cases} 0 & \text{wenn } n = 1\\ \frac{-\sum_{p=1}^{\Theta} Pr(M_C \in [M_P]) \cdot \log(Pr(M_C \in [M_P]))}{\log(n!)} & \text{wenn } n > 1. \end{cases}$$
(17)

Um Θ und C_P zu gewinnen, präsentieren Gierlichs et al. einen Divide and Conquer Algorithmus [3], der diese berechnet.

B. Informationstheoretische Metrik basierend auf Anonymitätswahrscheinlichkeitsverteilungen

Dass die Anonymitätsmenge einige Probleme mit sich bringt, zeigen Serjantov und Danezis [10]. Sie definieren die Anonymitätsmenge als Menge an Teilnehmern, für die es wahrscheinlich ist, eine bestimmte Nachricht gesendet zu haben. Anhand dieser Menge lässt sich die Anonymität bestimmen, die in einem Netzwerk gegeben ist. Ist ihre Größe 1, wird dem Teilnehmer keine Anonymität geboten, was den ungünstigsten Fall darstellt. Im günstigsten Fall jedoch ist sie die Größe des Netzwerks. In diesem Fall könnte jeder Teilnehmer die Nachricht gesendet haben. Ein Problem verdeutlicht die Anwendung des Pool Mixes, welches immer n Nachrichten speichert. So besteht für jede Nachricht, die das Mix Netzwerk durchlief, eine geringe Wahrscheinlichkeit, dieses noch nicht verlassen zu haben. Demnach muss jeder Sender von jeglicher Nachricht in die Anonymitätsmenge aufgenommen werden. Serjantov et al. [10] beobachten, dass sich die Anonymitätsmenge weder beim Rückkoppeln einer Nachricht, noch bei mehreren Nachrichten verändert. Damit begründen sie ihren Entschluss, Anonymitätsmengen seien in diesem Fall ungeeignet. Auch die sogenannte Kenntnisschwachstelle stellt die Verwendung von Anonymitätsmengen in Frage. Wenn im Beispiel von Abbildung 6 ein Angreifer wüsste, dass R eine Nachricht von A erhalten hat, könnte dieser rückschließen, dass E und S miteinander kommunizieren. Wenn also A, B, Coder D mit R kommuniziert und dies dem Angreifer bekannt ist, steht fest, dass die Senderanonymitätsmenge von S nur von Größe 1 ist. Auch in Mix Netzen mit größeren Grenzen als 2 kann die Größe der Anonymitätsmengen reduziert werden.

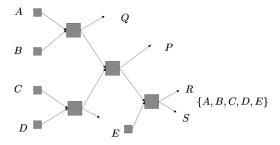


Abbildung 6. Schwachstelle von Anonymitätsmengen. [10]

Serjantov et al. [10] definieren ein Modell, das sich auf den mittleren Informationsgehalt bezieht, um dessen Qualität beschreiben zu können. Sie gehen von einem Modell eines Angreifers und einer endlichen Menge von Benutzern Ψ aus. Dabei ist $r \in R$ die Rolle eines Benutzers, nämlich $R = \{Sender, Empfänger\}$, bezüglich einer Nachricht M. U

ist dabei die A-posteriori-Wahrscheinlichkeit eines Benutzers $u \in \Psi$, der die Rolle r bezüglich M annimmt. Die effektive Größe S einer r Anonymitätswahrscheinlichkeitsverteilung U definieren sie gleich dem mittleren Informationsgehalt der Verteilung:

$$S = -\sum_{u \in \Psi} p_u \log_2(p_u) \tag{18}$$

Um die Wahrscheinlichkeitsverteilung an Anonymität zu berechnen, wird von einem gewöhnlichen Mix Netz ausgegangen, welches n Nachrichten mit einer Wahrscheinlichkeitsverteilung an Anonymität $L_0...L_{n-1}$, die aus Mengen von Paaren besteht, erhält. Des Weiteren sind alle Wahrscheinlichkeitsverteilungen an Anonymität für Nachrichten, die das Mix verlassen, gleich. Diese Verteilung A wird wie folgt definiert:

$$(x,p) \in A \text{ wenn } \exists i.(x,p') \in L_i \text{ und } p = \frac{\sum_{i.(x,p_j) \in L_i} p_j}{n}$$
 (19)

Auch die maximale Pfadlänge kann die Größe der Anonymitätsmengen verringern. Betrachtet man dazu Abbildung 7 und legt als maximale Pfadlänge 2 fest, so kann der untere Richtungspfeil, der im oberen rechten Mix mündet, nur eine Nachricht von C sein. Demnach wird die Senderanonymitätsmenge von S auf $\{A,B\}$ reduziert.

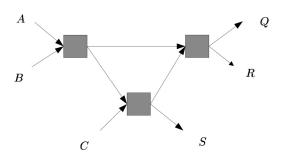


Abbildung 7. Reduzierung der Anonymitätsmenge durch maxinmale Pfadlänge. [10]

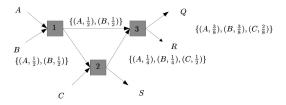


Abbildung 8. Reduzierung des mittleren Informationsgehalts durch maxinmale Pfadlänge. [10]

Ferner wird aufgrund der maximalen Pfadlänge auch der mittlere Informationsgehalt reduziert. Ist dem Angreifer aus Abbildung 8 die Pfadlänge von 2 bekannt, hat der Richtungspfeil von Mix 2 zu Mix 3 die Sender-Wahrscheinlichkeitsverteilung von $\{(C,1)\}$. Daraus folgt die Wahrscheinlichkeitsverteilung für Q oder auch R: $\{(A,\frac{1}{4}),(B,\frac{1}{4}),(C,\frac{1}{2})\}$. So wird der mittlere Informationsgehalt von 1.5613 auf 1.5 reduziert. Daher ist die Metrik von Serjantov et al. sowohl zum Vergleichen von Effektivität der Systeme, als auch zur Ermittlung der

Mächtigkeit von verschiedenen Angriffen geeignet.

C. Zonenbasierte Empfänger k-Anonymität

Wie Kelly et al. [4] beschreiben, richtet sich die zonenbasierte Empfänger k-Anonymität an die Sicherung der Lage des Empfängers. Eine Anonymitätszone AZ mit der Mitte x und Radius R_{AZ} wird von einem Sender für jeden Empfänger generiert. Die Nachricht wird von Benutzern zu einem Proxy weitergeleitet. Dieser überträgt die Nachricht zu allen Benutzern in der Anonymitätszone. Es existieren zwei Lösungen bezüglich der Anonymitätszone AZ, fix und lernfähig. Bei der fixen Variante nutzt der Sender eine initial großformatige AZ, nämlich $n_0 \gg k$. Dabei ist n_0 die Anfangsgröße an Benutzern in der Zone. Im Laufe der Zeit verlassen Benutzer diese Zone, daher ist das Ziel des Senders k oder mehr Benutzer in der Zone zu bewahren.

$$P\{n \ge k - 1\} = p(1 - \sum_{i=1}^{k-1} P\{n = i\})$$
 (20)

Dabei ist p die Wahrscheinlichkeit, dass der Empfänger in AZ bleibt und $P\{n=i\}$ die Wahrscheinlichkeit, dass i Benutzer in AZ bleiben. Anonymität ist gegeben, wenn $Pr[n \geq k-1] > \mu$.

Bei der lernfähigen Variante bestimmt der Sender die Größe von AZ, die k Knoten beinhaltet, anhand der Dichte an Benutzern. AZ wird im Laufe der Zeit basierend auf Mobilität erweitert.

$$R_{AZ}(t_1) = c(t_1 + t_0) - R_0 (21)$$

Dabei ist $R_0=\sqrt{\frac{k}{\Pi\rho}}$ der Anfangsradius, $t_0=-\bar{t_d}\ln(P_k)/k$ die Zeit, zu der k-Anonymität zu erreichen niedrig ist, und t_1 die Zeit, zu der der Radius größer ist. c ist eine Konstante R_0/t_0 und $R_{AZ}(t_1)$ der weitere Radius zur Zeit t_1 . Des Weiteren ist ρ die Benutzerdichte und $\bar{t_d}$ die durchschnittliche Benutzerzeit in der AZ. $P_k(t)$ stellt die Wahrscheinlichkeit dar, dass sich k Benutzer nach einer Zeit t in AZ befinden. Bei dieser Variante ist Anonymität gegeben, wenn $P_k(t)>\mu$ und R_{AZ} wächst.

D. Nachweistheorienanonymität

Die Nachweistheorienanonymität beschreiben Kelly et al. [4] als Maß für Wireless mobile ad-hoc Netzwerke. Dafür wird der Nachweiß anhand der Anzahl an entdeckten Paketen in einer gewissen Zeit bemessen. Dynamisch wird die Wahrscheinlichkeitsbestimmung für alle Paketauslieferungswege generiert. Darüber hinaus wird die Gesamtanonymität anhand der Anzahl an Bits mengenmäßig bestimmt. In Abbildung 9 lässt sich die Metrik erkennen, in der ein Angreifer dazu fähig ist, Pakete zu/von den Zonen h_1, h_2 und h_3 abzuhören und somit die Netzwerktopologie zu lernen.

Die Uneinigkeitsfunktion D(m) ist wie folgt definiert und findet als generalisiertes Anonymitätsmaß Verwendung:

$$D(m) = -\sum_{V \in F} m(V) \log_2(1 - \sum_{U \in F} m(U) \frac{|U - V|}{|U|}). \quad (22)$$

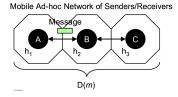


Abbildung 9. Nachweistheorienanonymität D(m). [4]

Dabei sind U und V geordnete Mengen der Benutzerkommunikationspfade und m(V) die Wahrscheinlichkeit einer wirkenden Kommunikationsrelation. $\sum_{U \in F} m(U) \frac{|U-V|}{|U|}$ klammert irrelevante oder widersprüchliche Nachweise aus. Hier ist Anonymität gegeben, wenn für einen vordefinierten Grenzwert δ $D(m) > \delta$ gilt.

E. Vereinigung von Netzwerk- und Anwendungsschicht

Clauß et al. [1] schlagen eine Vereinigung von Netzwerkund Anwendungsschicht vor, um unfangreiche Anonymitätsmetriken zu ermöglichen. Diesbezüglich bedienen sie sich einigen Entropien, wie beispielsweise der *Shannon-Entropy* und der *Rényi-Entropy*, die zunächst definiert werden. Die *Shannon-Entropy* stellt die durchschnittlichen Informationen einer Quelle dar:

$$H_{\emptyset}(P) = -\sum_{i=1}^{n} p_i \log_2 p_i$$
 (23)

So wird sie auch von Kelly et al. [4] definiert. Wenn ein festes Alphabet der Größe N gegeben ist, kann die obere Grenze H_{Max} der Entropie berechnet werden:

$$H_{Max}(P) = \log_2 N \tag{24}$$

Wenn eine Wahrscheinlichkeitsverteilung gegeben ist, kann die untere Grenze H_{min} der Entropie berechnet werden:

$$H_{Min}(P) = -\log_2 \max_P p_i \tag{25}$$

Dabei ist das Alphabet einer Quelle an Informationen gegeben durch

$$X = x_1, x_2, ..., x_n (26)$$

sowie eine Wahrscheinlichkeitsverteilung durch

$$P = (p(x_1), p(x_2), ..., p(x_n))$$
(27)

Clauß et al. verkürzen die Schreibweise zu

$$P = (p_1, p_2, ..., p_n). (28)$$

Ein weiteres Maß für den Informationsgehalt stellt die *Rényi-Entropy dar*:

$$H_{\alpha(P)} = \frac{1}{1 - \alpha} \log_2 \sum_{(X)} p_i^{\alpha}$$
 (29)

Um in der Netzwerkschicht die Entropie basierten Metriken anzuwenden, muss das Alphabet der Quelle angegeben werden. Möchte man den Sendern Anonymität gewähren, ist dies die Menge aller möglichen Sendern. Die Sicht des Angreifers bestimmt die Wahrscheinlichkeitsverteilung der Sender. Diese

Sichtweise ist die Wahrscheinlichkeit, der Sender einer Nachricht zu sein. Diese Entropie basierte Metrik sagt aus, wie viel Information dem Angreifer fehlt, um den Sender der Nachricht zu ermitteln. Bei Anonymität der Empfänger würden das Alphabet der Quelle und die Wahrscheinlichkeit entsprechend erstellt werden. In der Anwendungsschicht geben Benutzer Informationen über sich bekannt, um einen bestimmten Service nutzen zu können. Ein Benutzer wird also anhand seiner Attribute, die im Grunde alles sein könnten, repräsentiert. Jede Teilmenge dieser Attribute wird als Profil definiert. Diese Profile stellen das Alphabet einer Quelle der Entropie basierten Metriken dar. Die Wahrscheinlichkeitsverteilung, die ebenfalls benötigt wird, ist durch die Wahrscheinlichkeiten, mit der jedes einzelne Profil zu einer Teilidentität eines Benutzers gehört, gegeben. Um die beiden Schichten zu vereinigen, ist es notwendig, die Informationen der Netzwerkschicht als anwenderbezogene Attribute zu modellieren, um diese problemlos in die Profile der Anwendungsschicht integrieren zu können. An einer praktischen Umsetzung wird noch geforscht.

V. VERGLEICH DER METRIKEN

Untersucht man die verschiedenen Metriken auf Zusammenhänge oder Gemeinsamkeiten, lässt sich erkennen, dass einige der vorgestellten Metriken auf anderen aufbauen. So wird für einige Metriken als große Gemeinsamkeit deutlich, dass *k*-Anonymität als Grundlage genommen wurde.

Die Lücken von *k*-Anonymität veranlassen Machanavajjhala et al. [6] die Metrik *l*-diversity zu definieren, die Attributenthüllung vermeiden können soll. Auch für diese Metrik werden Probleme sichtbar, sodass eine neue Metrik, die sogenannte *t*-closeness von Li et al. [5] entwickelt wird. *l*-diversity und *t*-closeness unterscheiden die Verteilung des sensitiven Attributs in der Gesamtpopulation der Tabelle. So nimmt die Metrik *l*-diversity an, diese Verteilung sei keine öffentliche Information. *t*-closeness hingegen geht von einer öffentlichen Information der Verteilung aus. Dadurch kann der Beobachter weniger zusätzliche Informationen über spezielle Personen lernen.

Die ungeneralisierte Metrik von Edman et al. [2], das kombinatorische Anonymitätsmaß zählt die möglichen Sender einer erhaltenen Nachricht. Die generalisierte Metrik betrachtet stattdessen die Wahrscheinlichkeiten von jedem möglichen Sender, um eine Entropie-basierte effektive Anonymitätsmengengröße berechnen zu können. Aufgrund dieses Unterschieds ändern sich die Definitionen für das Anonymitätslevel. So bedeutet ein größerer Wert von per(A) im initialen Modell ein höheres Anonymitätslevel. Im generalisierten Modell hingegen ist das Anonymitätslevel am höchsten, wenn die Permanente minimal ist.

Der Ansatz von Gierlichs et al. [3] weist einen Hauptunterschied zu dem von Edman et al. auf. Sie betrachten die Verbindungen zwischen Sendern und Empfängern, nicht die zwischen Input und Output Nachrichten. Dadurch wird das anonyme Senden oder Empfangen von mehreren Nachrichten möglicht.

Bei der zonenbasierten Empfänger k-Anonymität von Kelly et al. [4] unterscheiden sich die beiden Ansätze um die Anonymitätszone AZ. Bei der fixen Variante wird eine initiale

großformatige Anonymitätszone genutzt, die sich aus der Anfangsgröße an Benutzern ergibt. Die lernfähige Variante unterscheidet die Größe der Anonymitätszone insofern, dass sie anhand der Dichte an Benutzern ermittelt wird.

VI. KONKLUSION

In diesem Paper wurden Metriken zur Datensatz- und Kommunikationsanonymität beschrieben. Datenmetriken finden Anwendung bei Daten oder privatem Inhalt, die in Datensätzen, meist Tabellen, veröffentlicht werden. Anonymisiert wird meist anhand Generalisierung und/oder Unterdrückung. Netzwerkmetriken haben die Anonymisierung von Kommunikationen zum Ziel.

Die Vergleiche der Metriken zeigen, dass einige Gemeinsamkeiten, aber auch Unterschieden zu finden sind. So wird für einige Metriken als große Gemeinsamkeit deutlich, dass k-Anonymität als Grundlage genommen wurde. l-diversity und t-closeness unterscheidet die Betrachtung der Verteilung des sensitiven Attributs in der Gesamtpopulation der Tabelle. Die Sichtweise der Verteilung als öffentliche Information von t-closeness schränkt Beobachter beim Lernen von zusätzlichen Informationen ein.

Ein bedeutender Unterschied ist im Ansatz von Gierlichs [3] et al. zum kombinatorischen Anonymitätsmaß zu verzeichnen. Anders als Edman et al. [2] werden die Verbindungen zwischen Sendern und Empfängern betrachtet, was das anonyme Senden oder Empfangen von mehreren Nachrichten ermöglicht.

Immer wieder werden nicht betrachtete Zusammenhänge der Metriken entdeckt, die Anlass zu Verbesserungen der Metriken oder Neuentwicklungen geben. Beispielsweise wird bei der Vereinigung von Netzwerk- und Anwendungsschicht noch an einer praktischen Umsetzung geforscht. Kontinuierliche Forschung ist demnach notwendig, um weitere nicht betrachtete Zusammenhänge ermitteln und überarbeiten zu können.

LITERATUR

- [1] Sebastian Claußand Stefan Schiffner. Structuring anonymity metrics. Proceedings of the second ACM workshop on Digital identity management - DIM '06, page 55, 2006. IV-E
- [2] Matthew Edman, Fikret Sivrikaya, and Bulent Yener. A Combinatorial Approach to Measuring Anonymity. 2007 IEEE Intelligence and Security Informatics, pages 356–363, May 2007. IV-A, 2, 3, 4, IV-A,
- [3] Benedikt Gierlichs, Carmela Troncoso, Claudia Diaz, Bart Preneel, and Ingrid Verbauwhede. Revisiting a combinatorial approach toward measuring anonymity. Proceedings of the 7th ACM workshop on Privacy in the electronic society WPES '08, page 111, 2008. IV-A, 5, IV-A, V, VI
- [4] Douglas J. Kelly, Richard A. Raines, Michael R. Grimaila, Rusty O. Baldwin, and Barry E. Mullins. A survey of state-of-the-art in anonymity metrics. *Proceedings of the 1st ACM workshop on Network data anonymization NDA '08*, page 31, 2008. II-D, II-H, II-I, 1, III-C, IV-C, IV-D, 9, IV-E, V
- [5] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-Closeness: Privacy Beyond k-Anonymity and 1-Diversity. Entropy, (2). III-A, V
- [6] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L -diversity. ACM Transactions on Knowledge Discovery from Data, 1(1):3-es, March 2007. III-A, V
- [7] Andreas Pfitzmann and Marit Köhntopp. A Proposal for Terminology. 2001. II

- [8] Pierangela Samarati. Protecting Respondents 'Identities in Microdata Release. *DisClosure*, pages 1–29. III-A
- [9] Pierangela Samarati and Latanya Sweeney. Protecting Privacy when Disclosing Information: k -Anonymity and Its Enforcement through Generalization and Suppression 1 Introduction. Main, pages 1–19. III-A
- [10] Aandrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In *Privacy Enhancing Technologies*. Springer, 2003. IV-B, 6, IV-B, 7, 8
- [11] Latanya Sweeney. A model for protecting privacy 1. *Ieee Security And Privacy*, 10(5):1–14, 2002. II-H
- [12] Traian Marius Truta and Bindu Vinay. Privacy Protection: p-Sensitive k-Anonymity Property. Diabetes. III-B, III-B

Censorship Resistant Publishing Systems

Julian Wälde

Abstract—P2P networks have proven to be very suited for creating large redundant and safe application platforms. Millions of users are using P2P protocols to share documents every day. Some content being shared on the internet is subject to censorship usually implemented by comparatively powerful adversaries. This paper also discusses how to design countermeasures that can help to achieve censorship resistance. We also give a quick overview over 3 Systems (Freenet, Publius, and Tangler) trying to achieve censorship resistance. In this paper we also show that the security assumption put into the verification process for file retrieval in Publius is very weak.

I. Introduction

In this section we will introduce the concepts of publishing systems and censorship as well as anonymity. In Section 2 we will explain the concept of censorship resistance. In Section 3 there will be an overview over a selection of techniques (Shamirs Secret Sharing, Bit Commitments, and Mix Nets/Onion Routing) we are going to need for the 4th section. In section 4 we will shortly describe 4 systems that are trying to achieve censorship resistance (Publius, Tangler, and Freenet). In section 5 we summarize the content and contributions of this paper.

Modern P2P networks have many desirable properties. Most of all they scale well and minimize the need for centralized infrastructure. Moreover they make it easy to employ redundancy and other safety related technologies. However, most designs for P2P networks do not address the security issues that arise from involving many parties (and their interests) into the operation of the now decentralized infrastructure.

P2P networks are widely used to spread copyright protected content, creating a genuine interest for the copyright holders to prevent the spreading of this content. Apart from distributing illegal content, P2P systems could also be used to spread other information that hurts the interest of other powerful parties in a more legitimate way. This makes resistance against censorship a possible design goal for new systems that aim at a user base that is looking to spread information of that kind.

We will mainly look at what we call publishing systems; systems that allow its users to appear in 2 roles. First publishers that have documents they want to distribute. And Second recipients, users that want to receive published documents using the system in question.

A. Censorship

Censorship is the effort to suppress the distribution of a document of some reason. In our context we are always looking at a situation in which the distribution of the document is already taking place, meaning the party interested in employing censorship can at best minimize the degree of distribution.

The most practical example for this is file sharing of copyright protected content. The copyright holder has a genuine interest (profit) to prevent an ongoing distribution of the content that he holds copyrights on. With this it often would also suffice to damage or tamper with the content being distributed. Still, publishing systems are far more valuable to people that want to publish material in an environment where they have to deal with government agencies as their adversary.

B. Anonymity and Censorship

Publishing documents that potentially will be subject to censorship usually is done by a few individuals that have come in possession of these documents. Since there are only a few people actually publishing sensitive documents it almost ever is a working strategy for the censor to use intimidation of the publisher to stop their publication. Some papers refer to this as "Rubberhose Cryptanalysis". The potential to employ censorship by means of violence against the publishers puts the designers of P2P publishing systems up to the challenge of protecting the identity of the publisher. It has also been observed that repressive regimes use extensive force against those interested in the censored documents. This adds sender and receiver anonymity to the list of desirable properties of a censorship resistant P2P publishing system.

II. CENSORSHIP RESISTANCE

In this section we explain the notion of censorship resistance. We give a set of properties that a publishing system should exhibit in order to ensure resistance to censorship. In section 3 we are going to explain techniques that are being used to achieve certain aspects of censorship resistance. In section 4 we will present 3 systems (Publius, Tangler, and Freenet) that are trying to achieve censorship resistance. In section 5 we will summarize the contents and contributions of this paper.

Before one starts to implement censorship resistance one might ask what would a perfect censorship resistant publishing system look like.

• Efficiency:

This requirement is a non functional one but we expect a good solution to perform well. A publishing system that does not scale well or performs very badly will tempt whistleblowers and their likes to use other possibly unsafe methods of publishing their material.

• Persistence of Information:

Documents published with the system are not to be changed by anyone but the publisher. Since this is clearly not possible to achieve we will adapt this requirement to what [8] describes as tamper evident. Meaning that if the

document is changed by anybody that is not the publisher the recipients will be able to detect the change.

• Publisher Anonymity:

As already mentioned intimidation might be used on those that publish documents that the attacker wants to censor. If it is possible to harass the publisher the system is very unattractive for publishing sensitive information.

Plausible Deniability:

Participants of the System should be able to plausibly deny knowledge of the content stored on computers owned by them. This way it is a lot harder for law enforcement agencies to intimidate them. The same goes for all contents of communication as well as for the fact that they participate in the system.

• Recipient Anonymity:

Even though it is not as important as anonymity for the publisher it is still thinkable that harassment on recipients is used to give them an incentive not to receive the documents being published using the system. This is especially important with copyright protected content because copyright holders actually used law-enforcement agencies to intimidate users of filesharing systems in order to create the impression that is not secure to use them.

• Resistance Against Denial of Service:

If it is feasible for an attacker to make the system seize working it doesn't matter how well the system is doing in all other parts since censorship will simply be implemented by turning the system off.

III. TECHNIQUES FOR IMPLEMENTING CENSORSHIP RESISTANCE AND ANONYMITY

In this section we are going to explain a selection of techniques that are useful to equip systems with certain properties that are vital to censorship resistance. The techniques presented in this section were selected for their occurrence in systems that will be introduced in section 4. In section 5 there will be an summary of this papers contents and contributions.

A. Shamir's Secret Sharing

Shamirs secret sharing [6] is a technique to expand a secret value into n pieces in a way that only $k \le n$ pieces are needed to recover the secret and that if any adversary that has only k-1 pieces get no information about the secret value at all.

In summary the secret s is encoded as an element of a finite field of suitable size. Then a random polynomial p with degree k-1 is chosen with p(0):=s. The n pieces then are the evaluation of the polynomial at chosen inputs that are not equal to 0 in the field. To reconstruct the secret one now takes k of the n pieces and uses Lagrange interpolation to recover p. p(0) gives the secret value.

Many approaches on censorship resistance use Shamirs secret sharing not for sharing secrets but to create plausible deniability for the operators of servers that will store possibly compromising content. Secret sharing is not used for any security but as incentive to participate in the system. Publius [9] is a good example for the usage of secret sharing to this

goal. Tangler [8] goes even further and uses it to generate inter document dependencies with secret sharing. This way Tangler makes it hard to delete single files without damaging others as well. The method used here is to make secret shares of documents the points that are used to derive the polynomials of new files.

B. Onion Routing

Onion routing is a generic method of implementing sender and or receiver anonymity. The basic concept of onion routing is source based routing that allows every hop only to know his predecessor and his successor without any additional knowledge about the rest of the path of a message.

If the messages include no information that can be used to deanonymize the sender of a message the receiver will have no idea where the message originated from. Onion routing can simply be put over any already existing protocol (that does not reveal the identity of the sender of a message within the message) for the purpose of securing anonymity.

Tor [5] is probably the most widespread instance of this technology. Apart from implementing receiver anonymity Tor also makes it possible to run any tcp bound server as so called "hidden service" obfuscating the location of the server (resp. its real address). In combination with any blog software this would make for a very simple yet almost complete instance of a censorship resistant publishing sytems. Unfortunately there are some very persistent performance issues with Tor that would make it extremly easy to stage an DoS attack against publishing systems that rely on it.

C. Bit Commitments

Bit commitments are a cryptographic protocol that can be understood as the following game. Player A wants to commit to a value s at a given point in time without revealing s in a way that he can later reveal s and that other players can be satisfied that player A has committed to this value before. To achieve this A has to create a commitment c that reveals nothing about the secret s in a way that it is computationally infeasible to create another s' so that c would be a satisfying commitment to s'.

Player A uses a publicly known collision resistant one way compression function H and publishes H(s) as his commitment. Once s is revealed other players can verify that player A has been honest.

Practical Implementations of bit commitments work with the assumption of the existence of one way functions. Given an one way function H one player commits to a value x by publishing H(x) (so called commitment). The commitment may be later revealed by publishing x. Given that x is chosen from a significantly larger set then the range of H it is impossible to say anything about x given only H(x). Therefore bit commitments can claim information theoretical security for confidentiality and complexity theoretical security against fraud (that is if one tries to cheat and reveal something different then the value they originally committed to).

IV. CENSORSHIP RESISTANT PUBLISHING SYSTEMS

In this section we are going to present 3 systems that aim at providing censorship resistance. Also we will give an argument against the security assumptions made by the authors of Publius [9]. In the last section we are going to give a quick summary of the papers contents and its contribution.

A. Publius

Publius [9] is a publishing system that tries to achieve censorship resistance introduced by Waldman et al. Publius assumes a fixed globally known list of Servers that will be used to store data. A user of the system that wishes to publish a document M will first encrypt the document with a random key. The key used to encrypt the document is now secret shared in n pieces s_i of which k will suffice to recover the key. Publius now employs a hash function H that basically is a truncated 64bit variant of MD5 [4] to hash the not encrypted document and one secret share at a time. The output of $H(M||s_i)$ is then used to select a server and as path on the server to store M and s_i . The publisher now publishes all hashes in an url and therewith all servers that contain data of his document.

If another user now wants to retrieve M he takes k of these n hashes visits the corresponding servers and retrieves the secret shares necessary to recover the key needed to decrypt M

As for verification this user now recomputes all hashes that he was originally given and is satisfied if he all secret shares he has retrieved give the corresponding hash values, that either the document has not been tampered with or that an attacker has found a collision for H.

Another scenario is that a publisher wants to alter or delete data that was previously published. Since it would be easy to implement censorship on Publius if everybody could alter or delete content that is stored on the server there is an authentication scheme that works as follows. A not publicly readable document (named password) is stored on the server that contains H(hostname||PW) with PW being a secret passphrase only known to the publisher.

B. Note On The Security Of Publius

As stated before the authors of Publius claim that breaking their scheme is at least as hard as finding collisions in their hash function ${\cal H}$ with

$$H(m) := top64bits(MD5(m)) \oplus low64bits(MD5(m))$$

This construction gives us a 64bit hash function an upper bound for the effort needed to find collisions for it. Using cycle detection it is possible to find 2 messages that collide for any given 64bit hash function using roughly 2^{32} operations¹ and a negligible amount of memory. Looking at the construction of H one can see that any attacker that has an efficient algorithm to find collisions for MD5 can easily construct collisions for H. [10] give us a upper bound of 2^{21} evaluations to find a collision in MD5 making it possible to find collisions for H within seconds. The author does not believe that this directly

break the tamper evidence of Publius but it shows that the sketch of a proof the authors of Publius gave in their paper does not give a convincing security assurance. Also there seems to be no real justification for using a 64bit hash function for Publius. And the issues that exist with MD5 speak for themselfes. Having a proper security proof for the file retrieval process using a rather modern hash function (e.g., SHA256) would create a lot more confidence in the security of Publius.

As for demonstration we present 2 colliding strings for the hash function described above (generated using the code in the appendix):

Input	H(input)	
"MSGbe7539826b88340f"	dbd7e33551911d55	
"MSG384dc40b4f0a0c94"	dbd7e33551911d55	

C. Tangler

Tangler [8] is another distributed (yet not really P2P) publishing system aiming at censorship resistance. It uses an approach that differs from that used for Publius a lot. Documents that are about to be published with Tanger will be split into equally sized (16384 byte) blocks. Each of these blocks is then "entangled" with other blocks chosen at random. The process of entangling is basically Shamirs Secret sharing but does not expand the data as much. Every block does not only hold it's 16K of data but also an x value that forms a (x, y) with all the polynomials p_i this block was involved in building $p_i(x) = y$. Every block of the document is now entangled (secret shared) n-1 already existing blocks as points to interpolate this block. The authors of Tangler believe that the inter document dependency created by this method creates a lot of redundancy since the owner of documents will duplicate data blocks that do not solely belong to their document.

All in all Tangler can be viewed as a distributed filesystem that employs this process they call entanglement on every block stored on the respective storage. Even inode² like structures are not excluded from this process since that would make it very easy to delete files (by deleting the inode). Every block is addressed by its SHA1 hash. The servers are organized in a CHORD [7] like ring only that every server has multiple node ids that change on a daily basis. A server with N node ids has

$$SHA1(K_A||[dN/14] - i)$$

with d being the number of days since 1th January 1970, K_A being the servers public key, and i being the numbers from 0 to N. This makes it relatively easy to add new servers and also to adjust the load of every server regarding it's capacities.

Tangler ensures that published files are not tampered with by having the publisher sign the root of a SHA-1 Merkle tree³ over the published blocks. Also Tangler employs measures to find and eject misbehaving participants from the system.

¹ for MD5 2³² evaluations take about 20 minutes on a intel core i5

²a part of the file system that uniquely identifies a file and contains information where to find the contents of the file

³a binary tree that has the property that every node is he output of a hash function applied to it's children. Originally introduced by Merkle [3]

D. Freenet

Freenet is an unstructured P2P network that is used for sharing files in a censorship resistant manner. In Freenet files are being identified by a key. The keys used for files are SHA1 [2] hashes of either a description of the contents or the contents itself. Before publishing the content (by making it present in the caches of other nodes in the system) the content is encrypted to give all caching nodes the incentive of plausible deniability. Every node in Freenet donates a portion of it's memory as cache to the system. Whenever a request for a document is successful every node that the document passes by will cache it in case there are future requests for it. If the cache of a node is full the least recently used documents will be dropped until there is enough space for the new document.

Retrieving files in Freenet is mostly a heuristic backtracking search on the graph of nodes in the network. A node that wants to retrieve data that belong to a given key will request that file from it's neighbors ordered by a distance metric of the routing key that is associated with the neighbor and the key searched for. Since every node works as cache for content that was looked up nodes end up storing (caching) data that belongs to keys close to their routing key. Also the more frequent a key is being looked up by different nodes of the network the more replicas of the corresponding value will exist. This results in popular content (that is more likely to be subject to censorship) being harder to censor then unpopular content. The caching also has the effect that nodes will over time become "good" in delivering request that are associated with their key since they store copies of everything they are being asked about. In addition to that, this also will result in very short lookup paths for popular content helping with potential load issues.

E. Routing Key Selection Protocol In Freenet

In order to achieve random key's for routing in freenet a special protocol is used. A node that wishes to enter the network selects a random number and commits to it via bit commitment. This commitment is then send to the node it uses to bootstrap it's routing table. The message containing the top commitment is now sent on a random walk of a given length. At each hop of the random walk the node that just received the message selects a random seed xor's it into the last commitment and commits to the result. At the end of the random walk all nodes of the path reveal the seeds they committed to. The monadic XOR of all these seeds is the routing key that will be used for the new node.

F. Dark Freenet

[1] suggests to run Freenet as a friend-to-friend network darknet. This creates the assumption that peers are only connected to peers they have a pre established trust relationship to. It is assumed that such a network would be a small world network allowing short paths between any given two nodes. The routing for such a network is a big challenge since unlike in normal P2P networks, nodes cannot simply select a set of neighbors that is well suited for routing. Since any node will have only very limited information about the overall

network [1] proposes that nodes will join the network with random routing keys and then adapt them in a way that routing becomes more efficient. The thought in mind here is that nodes that are close together should have similar routing keys. The proposed key space here is the real numbers in [0,1] with a circular distance (e.g. d(0.1,0.9)=0.2).

It has also been proposed to run nodes that operate as some kind of a bridge between both networks. Having a such gateway nodes would make a lot of additional content available in the dark net while most of the darknet still remains "dark" in a sense that the participants of Freenet will not see its nodes.

V. CONCLUSION

Censorship resistance is a desirable property for publishing networks once their users want to publish non trivial content. In this paper we have described what properties a publishing system should have to qualify as censorship resistance. We revisited a set of techniques that have been used trying to achieve censorship resistance. Also we have introduced 4 systems that try to achieve this goal using different approaches. In addition to this we have shown that the hash function used for Publius [9] does not have the properties the authors of Publius expected it to have.

REFERENCES

- Matthew Toseland Vilhelm Verendel Ian Clarke, Oskar Sandberg. Private communication through a network of trusted connections: The dark freenet. IV-F
- [2] P.E. Jones et al. US secure hash algorithm 1 (SHA1). 2001. IV-D
- [3] R.C. Merkle. Method of providing digital signatures, January 5 1982. US Patent 4,309,569.
- [4] R. Rivest. The md5 message-digest algorithm. 1992. IV-A
- [5] Paul Syverson Roger Dingledine, Nick Mathewson. Tor: The secondgeneration onion router. III-B
- [6] Adi Shamir. How to Share a Secret. 1979. III-A
- [7] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Computer Communication Review, 31(4):149–160, 2001. IV-C
- [8] M. Waldman and D. Mazieres. Tangler: A censorship resistant publishing system based on document entanglements. 2001. II, III-A, IV-C
- [9] M. Waldman, A.D. Rubin, and L.F. Cranor. Publius: A robust, tamperevident, censorship-resistant web publishing system. pages 5–5, 2000. III-A, IV, IV-A, V
- [10] Tao Xie and Dengguo Feng. How To Find Weak Input Differences For MD5 Collision Attacks. 2009. IV-B

```
APPENDIX
                                                       }
#include < stdlib . h>
                                                       printf ("precomp of hare finished %s at %l
#include <stdio.h>
#include <stdint.h>
                                                       pow = 0;
#include < string . h>
                                                       while (cir_h != beg_h) {
#include <bits/wchar.h>
                                                                pow++;
//#include "LED/LED.h"
                                                                if (pow > (n*2))
                                                                        findcircle(start++);
#define SIZE 100
                                                                getword(cir_h, cir_i);
#define ASIZE 52
                                                                getword(beg_h, beg_i);
#define LEN 19
                                                                cir h = hash(cir i);
                                                                beg_h = hash(beg_i);
char alphabet[] =
                                                       printf("%s %s\n",beg_i,cir_i);
"abcdefghijklmnopqrstuvwxyz"
                                               }
"ABCDEFGHIJKLMNOPQRSTUVWXYZ";
                                               int main(int c, char **v){
uint64_t hash(uint8_t *p) {
                                                       if (c > 1) printf("%11x", hash(v[1]));
        uint64_t h[2];
                                                       // findcircle (1337);
        MD5(p, LEN, h);
                                               return 0;
        return h[0]^h[1];
}
void getword(uint64_t x, uint8_t *p) {
        0[(uint64_t*)p] = x;
}
void findcircle(uint64_t start) {
        uint8_t beg_i[LEN];
        getword(start, beg_i);
        uint64_t beg_h = hash(beg_i);
        beg_h = hash(beg_i);
        uint8_t cir_i[LEN];
        uint64 t cir h = 0;
        getword(beg_h, cir_i);
        cir_h = hash(cir_i);
        uint64_t pow = 1;
        uint64_t n = 1;
        do {
                 if (n == pow) {
                         n = 0;
                         pow *= 2;
                         memcpy(beg_i, cir_i, LEN);
                         beg_h = cir_h;
                 }
                 n++;
                 getword(cir_h, cir_i);
                 cir_h = hash(cir_i);
        } while (cir_h != beg_h);
        cir_h = start;
        beg_h = start;
        int64_t i;
        for (i = 0; i < n; i++) {
                 getword(cir_h, cir_i);
```

cir_h = hash(cir_i);

P2P Service Delivery in Mobile Ad-Hoc Networks

Gregor Wicklein

Abstract—Mobile ad-Hoc Networks (MANETs) are a common research topic, as mobile communication devices becomes more and more available in our society. To provide services in such MANETs, peer-to-peer (P2P) delivery mechanisms are a common approach in this area, since their decentralized nature scales well on the needs of these networks. Besides the usual issues of P2Pbased service delivery, a MANET driven approach has to deal with numerous problems like frequent changes in the network environment, unstable connections and limited power-supply of the participating nodes. In this paper we give an overview of these issues and discuss some general approaches to deal with them. Several delivery-architectures, designed to address the special needs of MANETs, will be presented and compared. We also pay attention to the underlying routing-architectures, how they affect the service delivery mechanism and how both can be combined to work more efficient.

I. INTRODUCTION

Over the last years, P2P-based services became popular and are widely used. Starting with simple file exchange services, P2P architectures are nowadays used in various application areas like audio- and video streaming, Voice over IP, chatservices or digital payment systems. They all benefit from the advantages like failure-safety or better scalability of P2P based architecture, compared to centralized approaches. On the other hand, mobile devices like laptops, PDAs or smartphones established in modern society. Most of them are equipped with communication technologies (e.g. WiFi or Bluetooth), which allow them to form MANETs. These MANETs can be established spontaneously on demand. They can provide a platform for a P2P overlay structure to enable service delivery as mentioned above. The fact that such networks can be formed in absence of a central communication infrastructure make them attractive in military scenarios or crisis situations, where such infrastructure usually is not available. But also traditional consumer markets could benefit from MANET based service delivery. The rising popularity of social communities created a whole new area of social interaction-applications to share any kind of content. Since many of these applications are location dependent, they could scale well on a mobile ad-hoc network. Stieglitz et al. give a few examples of such applications and discuss the advantages of MANETs in combination to them [16]. But they also point out that real MANET communication, despite of available communication technologies, is rarely used in this kind of applications.

To address the mentioned issues, we will first give a short overview on the general architecture of a MANET. Afterwards, the technical challenges of service discovery- and delivery mechanisms in MANETs will be discussed. In the following sections two general service delivery approaches, namely *directory* and *directoryless*, are introduced. We will discuss both approaches in detail by presenting the current state of research in this areas. The last part of this paper analyses the underlying

routing protocols of MANETs and addresses their impact on the service delivery process. We introduce a few approaches to integrate the delivery mechanism into the routing layer and how this affects the efficiency of the MANET. We finish our paper with a comparison and discussion of the presented approaches and finally conclude our results.

To discuss service delivery in MANETs, we give a brief introduction into the topology of a MANET. A Mobile adhoc network consists of single *nodes*. Nodes are connected via wireless connections. Each of these wireless connections has a limited radius. To be part of the MANET, a node has to have at least one other node in its radius. Direct connections between nodes are called *single-hop connections*. Connections to to nodes outside the singe-hop radius have to be routed over other nodes to a link. Since the lack of a centralized routing infrastructure in MANETs, the nodes has to establish an autonomous routing algorithm. Routing in MANETs can be divided into three categories: reactive-, proactive- and hybrid routing:

- In *reactive routing*, routes are created on demand by flooding request messages across the network until they reach their destination.
- Proactive routing frequently sends routing updates over the network to ensure, that every node has up-to-date routing information. Proactive routing could establish connections faster than reactive, because it does not need to discover the path on demand, but has probably more overhead, since routing information has to be updated frequently.
- Hybrid routing tries to apply both approaches to increase
 efficiency. For example, a hybrid approach could divide
 the nodes into local groups to use the faster proactive
 routing inside the group and reactive routing for intergroup communication.

The applied routing approach in a MANET is defined by the underlying routing protocol, which is implemented on the routing layer. For the next sections we assume an underlying routing infrastructure, that allows *multi-hop connections* between all nodes. We discuss the impact of the routing protocol on the service delivery in detail in section VI.

II. GENERAL PROBLEM OF SERVICE DELIVERY IN MANETS

Besides the general problems of delivering services in P2P environments, a delivery platform for MANETs has to meet the requirements related to the technical conditions in a MANET. The two main issues related to mobile networks are:

 Limited power supply of single nodes. Each node has a limited power supply depending on its battery capacity. Since wireless radios are one of the most powerconsuming components in wireless devices, the amount of sent and received traffic has a direct impact on the availability of a node. In traditional wired P2P networks it is sufficient to guarantee that the bandwidth of the participating peers are not exceeded. However, in MANET based P2P infrastructures fair traffic splitting is more essential. Otherwise, heavier traffic load on a single node leads to an early breakdown of them and effects the long-term stability and availability of the whole network.

• Changing network conditions. MANETs are more affected by changing network conditions than classic networks. Since nodes are mobile, signal quality between them varies dependent on their changing positions and distances. This leads to a continuous change in the network topology. Also MANETs have a potentially high churn rate, since nodes join and quit the network, whenever leaving its signal range or turn of their radio to save energy. A service delivery process has to be aware of the fact, that a MANETs has only low guarantees on availability and stability of connections.

In the next sections we introduce different approaches for service delivery in MANETs. We pay attention on the techniques that are used to address the above mentioned issues.

III. SERVICE DELIVERY APPROACHES

Discovery and delivery for P2P services in MANETs is a complex task. First, we have to separate between consumer and producer. Producers are nodes, which provide at least one service to other nodes, and consumers are nodes, who intend to use at least one service. Services can be unique subjects, just provided by a single node, or redundant instances, provided by several nodes simultaneously. It is also common in P2P systems that nodes can take the role as consumer and provider at the same time. Ververidis defines the discovery mechanism as four basic functions [18]. Nodes can:

- · advertise a service
- query about services, provided by other nodes
- · select the most appropriately matched service
- invoke the service

To realize these functions, there has to be an advertise and request mechanism to share a service. In a centralized service delivery approach this is trivial, since there is just one single service provider, which nodes could request services from. In a P2P driven approach the task is more complex, since services are located on different nodes. That is why there has to be a way to globally look up the availability of a service and locate its potential providers. This task of resource management is a common issue in all kinds of P2P networks and has been addressed in different ways. Generally, we can divide all approaches in two categories: *Directory-based* and *directoryless*. We will discuss both categories in the following two sections.

IV. DIRECTORYLESS APPROACHES

The directoryless approach follows the maxim, that every node has to discover available services on their own. There is no kind of directory or source to manage available services and handle requests for them. A simple solution to this issue is, that nodes discover services on demand. Request messages are flooded to the network and any provider willing to approve the request answers the message. In a more proactive way, all nodes, which provide services, advertise them periodically. Other nodes can cache these advertisements locally and get a detailed view on all available services in the network. Also hybrid solutions are possible, where consumer only requests services reactive, whenever they find no appropriate provider in their local cache.

Since these approach does not require any centralized coordinator, it is appropriate for applying in in a MANET. Frequent joins and leaves or connection losses of single nodes have a lower impact on the stability of the whole network, since there is no single point of failure. On the other hand, permanent advertise and request messages produce a lot of overhead. This is a problem especially in MANETs. Every sent message, even if enough bandwidth capacity in the network is available, drains the nodes battery and effects its long-term availability. To reduce these network overhead, a lot of techniques have been created, like intelligent message routing, optimized multicasting or agents, who discover the network autonomously. We will now discuss a few of this approaches in detail.

A. Flooding

An example for flooding based approaches is Adder, a directoryless service discovery mechanism for MANETs in military environments [13]. In Adder, services can be reactive discovered or proactive advertised. Both modes are supported. If a node wants to advertise a service, it broadcasts its advertise message to all nodes in its one-hop radius. Each node, which receives this message, caches the service description and the address of the providing node. Afterwards, the node forwards the advertisement to all nodes in its one-hop radius and so on. Every advertisement has a hop-counter, which is decreased on every forward. This limits the propagation of the advertisement to a defined hop-radius around the provider. Since advertisements do not spread to the whole MANET, traffic is reduced, but, as a consequence, services are not accessible from all nodes in the network. For Adder, this effect is acceptable, since connection over a long multihop link produces a lot of traffic-overhead and service delivery over such long links are not desirable.

Besides the hop-counter every advertisement carries a *time-to-live value* (TTL). Services in the local cache of a node are just valid until their TTL is reached. Afterward, the providing node has to re-advertise its service to keep it available. This mechanism produces additional traffic, but is necessary to detect disconnected or moved provider nodes. The length of a TTL interval has to be chosen depending on the MANET conditions. Longer TTLs lead to lower traffic-overhead, but it takes longer to detect unavailable or moved provider. Shorter TTL reacts faster on changing conditions in the MANET and, therefore, are suitable for high dynamic MANETs with fast moving nodes. Faster responsiveness is provided on the cost of higher traffic overhead.

To avoid loops in the advertise mechanism, every advertisement message has a sequence number, which is increased on every re-advertisement by the provider. A node only updates its cache entry, if the sequence number in the received message is greater than its local one.

To request a service, a node initially checks its local cache for a service entry. If no none is found, it broadcasts a request message to its neighboring nodes. They, again, check their local caches and send, in case of a hit, a reply with the address of the providing node. If no entry is found, the node forwards the message to its neighbor nodes. Similar to an advertisement, every request message has a hop-counter to limit the search radius for a providing node. Since nodes can use their local cache to reply service requests of other nodes, a lot of overhead is saved.

An improvement in the request strategy is implemented into the Group-based Service Discovery Protocol (GSD), which works quite similar to Adder [2]. Services are advertised and cached locally on all nodes. Also hop-count-limits and TTL are included in this approach. The main difference is the way, how services are described. In Adder, services have an Universally Unique Identifier (UUID) paired with an ASCII-description. However, GSD uses the DARPA Agent Markup Language, an XML-based language to provide service descriptions [5]. A key feature of DAML is a group hierarchy. Each described service belongs to a defined group (See figure 1). The advertising of a service works the same as in Adder [13]. A providing node advertises a service and its neighboring node forwards the advertisement until its hop-count is reached. But every advertisement message carries an additional list of groups. If a node forwards an advertisement, it collects the groups of all service advertisements in its local cache. Then he adds these groups to the list in the advertisement and forwards it. If a node receives such a message, the list provides it the groups of all advertisement on all nodes on the path from the origin node. This information can be used to avoid traffic intensive broadcasts. If a node wants to request a service, it initially determines the group of this service. Then, it searches the group lists of every advertisement entry in its local cache for this group. For every match there is at least one node on the path to the origin of this advertisement, which holds a service advertisement from the same group. There is no guarantee, that this is the desired service, but it helps to reduce the search space. The request for the service is only sent on the paths, which have a potential candidate for our service. Only, if the requesting node does not receive any answers or has no matching entry for the desired service in its cache, it performs a broadcast. By this approach, the amount of needless traffic is reduced.

B. Agents

Another way to provide a directoryless service discovery are agents. Agents are autonomous instances, who move from node to node to collect and provide service information for them. An example for a MANET based discovery approach is the *self organizing dissemination and collection approach for MANETs (SORT)* [12]. In SORT, agents are used to proactive collect and deliver service advertisements across the network. If an agent stays at a node, it broadcasts a request to all

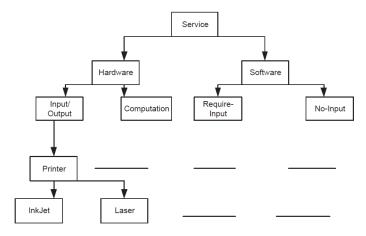


Fig. 1. DAML hierarchical grouping (Source: [2])

neighboring nodes to get their cached service information. After receiving all entries, it collects all of them, it is not aware of, and adds them to its own cache as well as the cache of the node, it stays on. After that, it selects a neighboring node and migrates to them to repeat the procedure. The selection of the succeeding node is essential for the efficiency of the agents. SORT provides two approaches for successor selection.

The Service Number-based (SN-based) approach selects the neighbor node with the lowest amount of new service information in relation to the amount of all new service information from all neighbors together. This is the node, that will receive the most new information and so benefits most from a migration to it.

The Moving Average-based (MA-based) approach compares the amount of service information every neighboring node has received for a specific period. The agent will move to the node with the lowest amount. This avoids, that nodes with many service information do not become updated, as it could happen in the SN-based approach.

An also important question, besides the migration selection, is, how many agents should observe the network, when they should be created and when destroyed. The number of agents has an impact on the efficiency of the service provisioning. Too many agents lead to a lot of traffic overhead in the network, but too few agents cannot provide the latest updates to the nodes in an acceptable time. In SORT, agents are created under two conditions. Whenever a service-providing node generates or updates a service or when a node does not receive a request from an agent for a defined period. Otherwise, services are destroyed again under two conditions. Whenever an agent detects another one on a neighboring node, it hands over its service information to the other one and destroys itself. An agent also destroys itself, if it has no other nodes in its neighborhood it could migrate to for a defined period of time.

V. DIRECTORY-BASED APPROACHES

The second essential approach for a service delivery mecanism is the directory-based solution. Some P2P service platforms are using a centralized directory instance to manage all services in the network. Service provider advertise their

services and consumer request a service only to this instance. For example Jini, a network architecture for distributed systems, uses a centralized look-up service to provide a register of available services [11]. Distributed hash tables are also a common approach to realize resource look-ups in P2P networks. Compared to the directoryless solution, this approach discovers services much faster, since only a single request on the directory is required. It is not necessary to frequently flood the whole network to advertise services or request one. The drawback on this approach is the delegation of the service coordination to a single or a small group of peers. To keep services discoverable, these nodes should always be available and accessible. In a traditional P2P network, such steadily available peers could be realized, but in MANETs we have no guarantee of availability of a single node. Additionally, the directory nodes have to expect a higher amount of traffic to handle advertisements and requests. As already discussed above, higher traffic on a single node leads them to faster power drain and earlier breakdown. Since it is a directory node its loss will have a critical impact on the network. This is why every directory-based solution in a MANET has to provide appropriate mechanisms to elect the directory nodes in respect of fair traffic sharing and has to support recovery mechanisms in case of a lost connection.

A. Central and distributed directory

To provide service discovery in MANETs by a directory-approach, the directory has to be reachable by every node in the network. One approach is to delegate the whole directory to a single node, which every other node could reach via multihop link. As discussed above, this directory node would be a single point of failure. In case of crash, the directory is lost and the network has to rebuilt it. Also the huge amount of potential long multihop connections to reach the directory node leads to high traffic overhead. In large scale MANETs the directory node could become the bottleneck of the network.

The opposite approach is to distribute the directory to a set of nodes, that could be reached by any node in the MANET via single-hop link. The smallest amount of nodes in this case is defined by the minimal dominating set of nodes. The fact, that no multihop connections are necessary to request a service from the directory, decreases the traffic overhead in the network. Otherwise, the minimal dominating set does not accomplish the needs of a MANET. On the one hand, it is hard to calculate without a global scope on the network, on the other hand, MANETs continuously change their structure caused by the mobility and the frequent joins and leaves of nodes in the network. This also leads to a continuous change of the dominating set and as a result to a high coordination overhead. Also fair traffic sharing is not taken into consideration, which could lead to traffic overload on single nodes.

B. Clustered approach

A compromise to both approaches is the cluster architecture. The network is clustered into separate subnets. Each subnet has a coordinator for handling internal routing and to communicate with other subnets in the cluster. An example for this approach

is the *Adaptive Service Discovery Protocol* from Jayapal et al. [7]. As mentioned before, the nodes become grouped into subnets. To scale well on the MANET architecture, the nodes in this subnet have to be in the same physical area to perform internal routing most efficient. Therefore, the nodes form their subnet and elect their leader on their own by performing a vote. Each node calculates a value, based on its battery value and its distance to the center of the proposed network. The node with the highest value becomes the local coordinator (LC). Whenever the LC's calculated value drops below a defined threshold or the LC disconnects, the vote becomes proceeded again to elect a new one.

To communicate between different subnets, a global coordinator (GC) gets elected by the same voting algorithm used on LC election. The GC has to be known only by the LCs. An LC stores, besides a link to the GC, a table of all available services in its subnet. Every node in the subnet, that wants to provide services, has to advertise its service descriptions to its LC. On the other hand, nodes can query for a service by sending a request to their LC. The LC checks its local service table for an existing provider inside the subnet. If one is found, the request is directed to the node, which hosts the desired service. If no local service provider is found, the LC directs the request to the GC. The GC holds a table with all available services and the corresponding LC for the service. If any of them matches the request, the request is forwarded to the appropriate LC and from there to the providing node. If a service is provided by multiple nodes, both LC and GC choose the one with the lowest workload and the lowest distance to the requesting node to reduce traffic and provide fair work sharing.

The introduced approach clusters the network to benefit from fast service discovery inside single subnets and global service availability by providing a global discovery mechanism. Also aspects, like limited battery power and fair work sharing, are taken into account.

One of the drawbacks of the approach is the heavy load on the LCs. Since they are responsible for service coordination and global communication, they have a double burden. Kim et al. introduced a way to address this issue by splitting the role of the service coordinator and the LC [9]. Their fundamental concept of dividing and coordinating the network is similar to the one presented by Jayapal et al. The main difference is the service directory management. The LC, here referenced as backbone node, acts also as service directory and receives advertisements from provider nodes. But to reduce traffic on the LC, the LC caches the service directory on a node in its subnet. This node, called proxy-directory, is responsible for handling service requests from other nodes in the subnet. Furthermore, the proxy-directory does not stay on a static node. To divide the work equally, the directory hops from node to node on a defined frequency. Since all nodes in the subnet are informed on the frequency and the hopping order, each node can calculate the actual position of the proxy-directory, based on the current time. It is not necessary to inform each node on a directory-hop, which would produce a lot of traffic. The permanent directory movement also allows fast recovery on a lost proxy. The LC, in this case, caches the directory to the next node in order and inform the other nodes about the change.

C. DHT-based approach

We saw that all above introduced approaches [7] [9] based on the idea to cluster the network and divide the service directories onto the single subnets. Another common approach to distribute a directory for service discovery in P2P networks is a *Distributed Hash Table (DHT)*.

The *Peer-to-peer Overlay STructure (POST)* is an example for a DHT-based solution, which realizes a robust service directory in a MANET [3]. Like before, the MANET is divided into subnets, called zones. Again, it is necessary that these virtual zones are chosen fitting to the physical structure of the MANET, which means that nodes in a virtual zone should be physically close together. To provide a distributed service directory, each zone in the network is assigned to a subset of keys from a defined key space. In each zone, a set of nodes provide the zone virtual manager service (ZVMS). These nodes keep track of physical location of all other zones in the MANET and which keyset is mapped to them. Nodes, which want to provide or request some services, need only the address of their local ZVMS and a global uniform hash function.

To advertise a service, the providing node generates the hashvalue on its service description. Then, the node sends this hashvalue to their local ZVMS. The ZVMS calculates, based on the received hash, the physical position of the zone, which is responsible for storing the directory entry for the service and returns it to the node. Finally, the advertising node sends the hash, the service description and its address to the calculated zone. Inside this zone, a defined set of nodes is responsible for storing the service advertisement. The storing algorithm is designed to restore a service information, even if just fragments of the stored values are available. This allows the directory to stay intact, even in case of a disconnected node, which is part of the distributed directory.

The request of a service works similar to the advertisement. The requesting node generates a hashvalue of the service description, it is interested in. The local ZVMS informs it about the responsible zone, whereto the requesting node sends its request. In case of a match, the responsible directory node in the zone answers the request with a list of nodes, that provide the desired service.

As we saw, the approach, introduced by POST, has similarity with those from Kim et al. and Jayapal et al., like dividing the MANET into subnets and usage of a local coordinator to coordinate global communication. Certainly, the DHT-approach differs from the other ones by the fact, that it does not need any kind of global coordinator to provide global service discovery. Every service advertisement is stored at a defined position, which can be calculated out of the service description itself. The lack of a global coordinator means a lack of a single point of failure, which increases the robustness of the approach. Otherwise, of fact, that a service advertisement has to be stored at a defined place could lead to an inefficient directory look-up process. Mention a situation, where a node

needs a service and another node in the same zone provides this service. In Kim's and Jaypal's approach, the request could be handled inside the zone, since the producing node advertises it to the local directory. In a DHT, the advertisement of the service is stored based on its hashvalue, which could be at the opposite end of the MANET. In this case the requesting node has to forward its request across the whole MANET just to invoke a service on a node right next to him. For those situations, the previously introduced approaches would outperform POST in case of look-up efficiency.

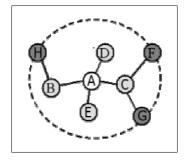
VI. CROSSLAYER SOLUTIONS

All solutions introduced in section IV and V are designed to operate in an multihop wireless network. To provide service delivery, they assume the presence of an multihop routing protocol to communicate. Following the ISO/OSI model, the routing process should operate on the routing layer, whereas service delivery mechanisms are typically placed on the application layer. Otherwise, both processes, routing and service delivery, share similar routines. For example, the routing protocol has to discover nodes to establish links between them, whereas the service platform has to discover services on nodes to invoke them. But, because both mechanisms operate on different ISO/OSI layers, we can not combine them to reduce traffic. A simple solution is to violate the concept of layer separation and integrate the service discovery and delivery mechanisms into the routing mechanism. This approach was first introduced by Koodli et al. [10]. They proposed a proactive routing protocol where the service advertisements are piggybacked onto the routing packets. Since their publication, a lot of research has been focused on these, so called crosslayer solutions. In this section we describe such crosslayer solutions in detail. We give examples how both, directory and directoryless service discovery approaches can benefit from routing layer integration.

A. Directoryless approaches

An example for a crosslayer service discovery protocol is *E-ZRP*, designed by Ververidis et al [17]. They use the *Zone Routing Protocol (ZRP)* and extend it with service discovery mechanisms [4].

ZRP is a hybrid routing protocol designed for MANETs. Every node uses a proactive routing approach for all nodes inside a zone with a defined radius around him (See Figure 2a). Outside of this zone, routes are established in a reactive way. Therefore, ZRP defines two sub-protocols, the Intra Zone Routing Protocol (IARP) and the Inter Zone Routing Protocol (IERP). To provide proactive routing, a node periodically contacts its neighbors for their presence. These requests are routed until an n-hop distance, so the node gets informed about all nodes in this zone and saves this information to its local IARP-table. To establish a connection to another node inside the local zone, the node can now use this table to calculate the route. For connections to nodes outside of the local zone. a node uses IERP. To request a route with IERP, the node bordercasts its request. Therefore it sends its request message to all nodes at the border of its zone. The bordernodes check



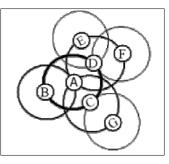


Fig. 2. a) zone with zone radius = 2, b) illustration of bordercasting (Source: [14])

their IARP-table for a matching entry and send a reply in case of a hit. If the bordernodes do not find a matching entry, they bordercast the request to their bordernodes (See Figure 2b). This mechanism is much more effective with regard to traffic overhead, than simply flooding the network with requests.

E-ZRP extends ZRP by adding service discovery functionality to the proactive part of the protocol. Services in E-ZRP are identified by UUIDs, which means that the potential services have to be known a priori by all nodes. If a node wants to provide a service, it piggybacks the UUID onto every IARP-message. By this, every node has, besides routing information, the UUIDs of all available services from other nodes in the zone in its IARP-table.

In a later work, Ververdis et al. extended their work by introducing the Adaptive SerVicE and Route Discovery Pro-Tocol for MANETs (AVERT) [18]. AVERT follows the E-ZRP approach to extend ZRP with service discovery mechanisms, but also includes some extensions to improve efficiency and performance. ZRP uses by default a fixed zone radius for all nodes. It is difficult to determine an adequate radius at setup, since it depends on the structure of the MANET and the mobility of the participating nodes which is not a-priori known. Also, since MANETs are highly dynamic, the range for a good value could change permanently. AVERT addresses this problem by using Independent Zone Routing (IZR), a sophisticated version of ZRP. It differs from classic ZRP by the ability to dynamic change the zone radius based on the actual traffic conditions. Initially, IZR measures the traffic (IERP and IARP combined) on a node for a period of time. After that period, the zone radius is increased. The traffic gets measured again on a time period. If the amount of traffic decreased, the zone radius becomes again increased and the procedure repeats. If a higher amount of traffic is measured in this period, the zone radius will be decreased and the procedure repeats but this time with a decreasing zone radius. By this, the best radius with the minimum amount of traffic can be determined. After this initial process, the traffic is monitored continuously and and the zone radius becomes corrected, whenever is necessary.

Another issue with crosslayer approaches is the size of the service descriptions. Since these descriptions are piggybacked onto the routing packages, they have to be as small as possible to keep the packages compact. Many service delivery platforms using XML-based descriptions (e.g. DAML) to identify services. But, since XML is not optimized on small

data amounts, it is not appropriate to be piggybacked onto a routing packet. E-ZRP and AVERT are using UUIDs to identify a service. Thus, every service can be described as a singe number and it is possible to transfer this number in routing packages. The drawback of UUIDs is, that every possible service has to be known a priori by all nodes in the MANET. It is not possible for a node to advertise new kinds of services. Therefore UUIDs are poor in terms of flexibility and scalability.

To address this problem, Outay et al. introduced *BF-SD-ZRP* [14]. Like E-ZRP and AVERT, BF-SD-ZRP extends the ZRP protocol with service discovery mechanisms. It differs from the previous approaches by using *bloom filtering* for service descriptions.

Bloomfilters are probabilistic data structures optimized on space-efficiency. With those filters, every node in BF-SD-ZRP can advertise all its services together in a single 128-bit vector, called bloomfilter. These vectors become advertised via IARP-packages, similar to E-ZRP and AVERT. If a node want to know, whether another node provides a particular service, it checks its corresponding bloomfilter. Since it is just a probabilistic data structures it is possible to get false positive results on a check (but no false negatives). This could lead to a service invocation on a node, which can not handle these request. In this case, the node simply denies the request, which has no further impact on the system. Therefore false positives are no problem, since their rate of occurrence is in a tolerable range.

Another drawback of bloomfilters is, that they carry no information which services exactly are provided by a node. They can just determine, if a node possibly provides a particular service or not. Therefore it is not possible to get a global view on all available services in the MANET.

The advantages of bloomfilters are their flexibility and scalability. Service advertisements are not bound to a predefined set of service classes. Also service descriptions can be of any size and complexity but will always result in a singe 128bit vector.

B. Directory approaches

The above presented crosslayer approaches have in common that they extend ZRP [4] to provide service discovery mechanisms. Since ZRP has no central coordination instance and no central routing directory, the protocol could be categorized as directoryless in terms we discussed in section IV and V. Therefore, the crosslayer approaches based on ZRP could also be referred as directoryless.

But also directory driven approaches could benefit from routing layer integration. In section V, we introduced POST [3], which uses a DHT to provide a service delivery platform. To show the impact of crosslayer techniques on directory approaches, we present *Ekta*.

Ekta operates, like POST, on a DHT, but differs from it by integrating this functionality into the routing layer [6]. Therefore it uses, a routing overlay for the providing of DHTs [15].

Pastry uses a circular key space and every node has a 128-bit nodeID. A key/value pair in the DHT is stored on the

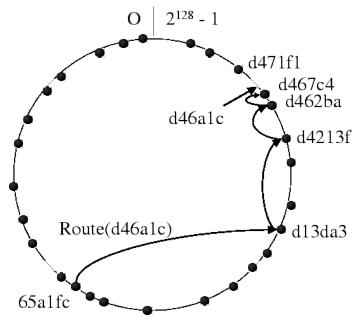


Fig. 3. Routing a message from node 65a1fc with key d46a1c (Source: [1])

node, which nodeID is numerical closest to the key. To reach a node, which is responsible for a key, pastry provides a routing mechanism. Every node has a routing table. Every entry in the routing table represents a part of the key space and has the address of at least one node with a nodeID in this key space part. By design, every node that receives a request with a key either has the numerical closest nodeID to this key and is responsible to handle the request or otherwise forward the message to a node, which nodeID is numerical closer to the key then the nodes own. Figure 3 illustrates this routing process by delivering a message to the desired destination.

Pastry guarantees that in a network with N nodes, a message reaches its responsible node in less than $\log_b N$ hops (where b is a configuration parameter). Pastry is commonly used in the internet. Therefore, the protocol is implemented on application layer and the nodeIDs are mapped to IP-addresses. To scale better on a MANET, Ekta implements Pastry on the routinglayer.

To provide multihop communication in the MANET, Ekta uses *Dynamic SOurce Routing (DSR)* [8]. DSR is a reactive routing protocol. To establish a link between two nodes, it floods the network with request packages, which are forwarded by other nodes until they reach their destination. The destination node sends a reply message to the sourcenode, that contains the necessary routing information to establish the link. To reduce overhead, all nodes cache the routes, that could be extracted from received request messages.

Ekta combines DSR and Pastry to one routing protocol, to work most efficiently. As mentioned before, every entry in the Pastry routing table presents a part of the keyspace and has at least one nodeID of a node is this keyspace. Ekta extends this table, by adding the route to the corresponding node of each nodeID, as far as a route is available in DSR-cache. The routes are permanently updated, as soon as new request messages with new routing information are received. If

Pastry needs to forward a message, it picks the corresponding entry from the routing table and forwards the message to a node, which is listed in this entry. The benefit of the additional routing information is, that Pastry could choose the node with the shortest route (respectively lowest hopcount) as target. This additional routing information is only available on the routing layer and this explains, why Pastry benefits from the implementation as a crosslayer solution.

VII. DISCUSSION

In the previous three sections, we introduced fundamental design approaches for service discovery and delivery platforms and presented related research in this area. We will now compare those approaches against with each other to discuss their advantages and drawbacks.

A. directory vs. directoryless approaches

In section IV we introduced directoryless approaches, whereas section V focused on directory-based service discovery. In general, none of those approaches is clearly superior. Since MANETs are highly dynamic systems, robustness is an important attribute of them. This means, a service delivery platform should be available, even in case of disconnected or moved nodes or other changed network conditions. By design, directoryless approaches perform better in this area, since they have no single point of failure. However, directory based approaches have addressed this problem by providing techniques, like redundant directories or efficient recovery mechanisms.

Other important attributes are traffic overhead and delay. Traffic overhead describes the amount of traffic, that is used for service provisioning functions, like advertising, requesting and invoking of services, whereas delay describes the time, that is necessary to discover and invoke a service. For both metrics, neither directory nor directoryless approaches generally outperform the other one. It relies on the MANET conditions, like the number of nodes, the mobility of them, the number of advertised services, the number of invocations and the consumer/producer ratio.

For a directory driven approach, service advertising and requesting produce a similar amount of traffic and a similar delay, since both, producer and consumer, have to contact the service directory. For a directoryless, it depends on the discovery design and the consumer/producer ratio. In proactive delivery platforms, a service advertisement has to be flooded around the network to inform all nodes about the advertisement. Therefore, service-advertisement generates a lot of traffic and takes long, whereas service request can be handled by the nodes local cache and generates no traffic and a minimum delay.

Assuming that we have an directoryless service delivery platform, which discovers services in a proactive way. May there be a lot of consumer, who frequently request services and just few producers, who rarely advertise new services. In this case the directoryless approach performs well, with minimal service request delay and little traffic overhead. In this scenario it could outperform the directory-based approach. Otherwise,

in a MANET with more producers and fewer consumers, the proactive directoryless approach would perform bad. In this case a reactive discovery mechanism would suit better.

Also the underlying routing protocol has an impact on the service delivery. It is clear, that a proactive discovery mechanism performs better on an proactive routing protocol than on a reactive. But since many directoryless approaches are not clearly proactive or reactive, but use a hybrid mechanism, it can not generally be figured out, which underlying routing architecture suits best. Statistically, for both, directory and directoryless service discovery architectures, proactive and hybrid routing architectures mostly perform better than reactive ones.

B. application vs. crosslayer solutions

In section VI we introduced how service discovery and delivery could be handled on the routing layer by integrating this mechanism into the routing protocol. All presented approaches could evaluate, that they generally perform better than pure application layer approaches. In Ekta [15], for example, the authors examined this fact in detail by implementing their approach as both, crosslayer and applicationlayer platform. Their simulation shows, that the crosslayer outperforms the applicationlayer implementation in case of delay and traffic overhead.

The problem with such integrated protocols is the violation of the ISO/OSI layer separation. This leads to further problems. Changes in the routing protocol could make it incompatible to the original version. In special purpose situations with a homogeneous node structure, this could be acceptable. Think of a crisis scenario, where rescue teams use special communication devices, which are optimized on a single communication configuration. In this case, a crosslayer solution is suitable, since these devices are not designed to interact with other communication infrastructure.

Otherwise, in a scenario where only a subset of nodes of a MANET want to use a crosslayer service platform, the modified routing protocol could lead to incompatibility. But with the growing number of wireless devices and the different kinds of possible services, the adherence of routing and communications standards becomes more important.

VIII. CONCLUSION AND FUTURE WORK

In this paper we discussed P2P service discovery and delivery platforms which targets on usage in MANETs. We figured out the special requirements of a MANET driven P2P system, compared to traditional wired solutions. Two general design approaches have been introduced, the directory and the directoryless. For both, related research has been presented and discussed, how they can be optimized to perform well in a MANET environment. We also payed attention to so called crosslayer solutions, which combine both, routing and service discovery into one protocol. In the last section, the introduced approaches were compared and discussed, as in which scenario they fits best. As we have seen, there is no general fitting design approach for a service delivery and discovery platform

in a MANET. Every decision has to be made with respect to the target platform, target environment and target services.

The area of MANET driven P2P service delivery has been in focus of research for over a decade. Although wireless communication devices became more and more common, MANET based systems are still rarely used in everyday life. Therefore, further research in the area of P2P service delivery in MANETs is still necessary to meet the requirements of the technological progress in the years ahead.

REFERENCES

- M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D.S. Wallach. Secure routing for structured peer-to-peer overlay networks. ACM SIGOPS Operating Systems Review, 36(SI):299–314, 2002.
- [2] D. Chakraborty, A. Joshi, Y. Yesha, and T. Finin. GSD: A novel group-based service discovery protocol for MANETS. In *Mobile and Wireless Communications Network*, 2002. 4th International Workshop on, pages 140–144. IEEE, 2002. IV-A, 1
- [3] Anandha Gopalan and Taieb Znati. POST: A Peer-to-Peer Overlay Structure for Service and Application Deployment in MANETs. Structure, pages 1006–1015, 2005. V-C, VI-B
- [4] Z.J. Haas, M.R. Pearlman, and P. Samar. The zone routing protocol (zrp) for ad hoc networks. 2002. VI-A, VI-B
- [5] J. Hendler and D.L. McGuinness. The darpa agent markup language. IEEE Intelligent systems, 15(6):67–73, 2000. IV-A
- [6] Y Charlie Hu and West Lafayette. Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks. *Computing Systems*, (Wmcsa), 2004. VI-B
- [7] Cynthia Jayapal. Adaptive Service Discovery Protocol for Mobile Ad Hoc Networks. European Journal of Scientific Research, 49(1):6–17, 2011. V-B, V-C
- [8] D.B. Johnson, D.A. Maltz, J. Broch, et al. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. Ad hoc networking, 5:139–172, 2001. VI-B
- [9] Sok-hyong Kim, Joo-young Baek, Seung-hoon Lee, and Young-joo Suh. Directory-Hopping based Service Discovery for Dynamic Large-scale Wireless Networks. Science And Technology, 2009. V-B, V-C
- [10] R. Koodli and C. E. Perkins. Service discovery in on-demand ad hoc networks. ETF Internet Draft, 2002. draft-koodli-manet-servicediscovery-00.txt. VI
- [11] Sun Microsystems. Jini architecture specification, Nov. 1999. V
- [12] Tomoyuki Ohta, Takeshi Hashimoto, and Yoshiaki Kakuda. Self-Organizing Real-Time Service Dissemination and Collection Using Mobile Agents for Mobile Ad Hoc Networks. In 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops, pages 199–206. IEEE, March 2011. IV-B
- [13] G Oikonomou, I Phillips, and a Grigg. ADDER: Probabilistic, Application Layer Service Discovery for MANETs and Hybrid Wired-Wireless Networks. 2011 Ninth Annual Communication Networks and Services Research Conference, pages 33–40, May 2011. IV-A
- [14] Fatma Outay, Florent Kaisser, Veronique Veque, and Ridha Bouallegue. BF-SD-ZRP: A Smart Integrated Scheme for Service and Route Discovery in Mobile Ad Hoc Network. 2010 IEEE 12th International Conference on High Performance Computing and Communications (HPCC), pages 404–412, September 2010. 2, VI-A
- [15] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware* 2001, pages 329–350. Springer, 2001. VI-B, VII-B
- [16] Stefan Stieglitz and Christoph Fuchß. Challenges of MANET for Mobile Social Networks. *Procedia Computer Science*, 5:820–825, January 2011.
- [17] C.N. Ververidis and G.C. Polyzos. Extended ZRP: a routing layer based service discovery protocol for mobile ad hoc networks. The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, pages 65–72, 2005. VI-A
- [18] C.N. Ververidis and G.C. Polyzos. Service discovery for mobile ad hoc networks: A survey of issues and techniques. *Communications Surveys* & *Tutorials*, *IEEE*, 10(3):30–45, 2008. III, VI-A