OverlayMeter: Robust System-wide Monitoring and Capacity-based Search in Peer-to-Peer Networks

Inaugural-Dissertation

zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Heinrich-Heine-Universität Düsseldorf

vorgelegt von
Andreas Disterhöft
aus Duschanbe in Tadschikistan

Düsseldorf, März 2018

aus dem Institut für Informatik
der Heinrich-Heine-Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

Berichterstatter:

1. Jun.-Prof. Dr.-Ing. Kalman Graffi

2. Prof. Dr. Michael Schöttner

Tag der mündlichen Prüfung: 24.09.2018

Abstract

In the last decade many peer-to-peer research activities have taken place. Applications using the peer-to-peer paradigm are present and their traffic, depending on the region, accounts for a significant proportion of the total traffic on the Internet. The defined goal of the systems is to deliver a certain quality of service, which is a challenge in decentralized systems. This is due to the fact that participants have to make decisions based on their locally available information. In order to make the best decisions, a solid and extensive data basis is indispensable. For this purpose the literature on the field of p2p networks proposes monitoring the system, an approach that we follow in this work. Monitoring refers to the gathering and dissemination of system- and peer-specific data. This dissertation deals with open research questions for the improvement and extension of monitoring approaches. Furthermore, issues to simplify procedures for putting such peer-to-peer systems into operation are addressed in this work.

In the first part we deal with monitoring procedures in the system-specific context. Here, we focus on tree-based procedures, as we have seen the biggest potential for this class, and tackle existing problems in their robustness. State-of-the-art tree-based approaches proposed in the literature are said to be highly precise but not robust, which is due to the dynamics of the users. As a consequence communication paths in the monitoring tree are disturbed, interrupted and in the worst case they have to be reconstructed. This leads to data loss which has a negative influence on the precision of the monitoring. Our answers to open research questions include the proposal of redundancy in a smart distribution function and the proposal of additional mechanisms that improve the perception of each participant in the monitoring structure. Thus, we present two methods that significantly increase the robustness, keep the precision on a consistently high level and have a good cost-benefit ratio.

In addition to the improvement of existing monitoring systems for system-specific data, we treat phenomena that lead to an impairment of the monitoring result. The influence of malicious participants in monitoring procedures is not adequately analyzed in the literature and existing solutions for the general handling of such participants are complex and could open doors for further attack vectors. We analyze the influence of malicious participants who manipulate the result or do not adhere to protocol specifications, highlight the serious vulnerabilities, and present mechanisms for mitigation. In a comprehensive evaluation, we show that manipulation attacks can be limited to a convex hull based on outlier detection and attacks on the monitoring structure can be reduced by verifying their origin. Overall, the proposed extension for tree-based monitoring systems operates passively and is therefore versatile. On the other hand, we are pursuing a undisclosed field in monitoring; the incomplete participation of users in monitoring solutions. We identify the impact that such a behavior has on the accuracy of the monitoring and present a solution solving this problem. We focus on collecting the global state of the system and are therefore interested in the monitoring information of all participants. As a solution, we propose a generic middleware that reuses existing monitoring procedures. Here, we rely on an organization of the active participants in the middleware to measure the passive participants by probing them and feed captured information into the monitoring solution. The evaluation shows a high degree of precision, which among others depends on the precision of the probing methods.

Moreover, we address the peer-specific data gathering and the capacity-based peer search. The goal is to create an efficient indexing structure that efficiently stores highly dynamic

peers' heterogeneous capacities in a distributed manner. We aim on an accurate peer search mechanism that obtains peers fulfilling set requirements. Such systems are motivated with the delegation of tasks and the realization of distributed computing in a decentralized context. The literature presents techniques that do not take all peers' capacities into account during the search process, do not scale with the number of capacities or do not comply with the requirement of storing highly dynamic data. We propose two solutions that efficiently store highly dynamic peer capacities and distribute the load fairly among all participants. While our first solution globally sets the number of capacities to a constant, our second solution uses a dynamic approach. The proposed search processes perform quickly and precisely, so that they meet the demands.

In the final part of this dissertation, we focus on a barrier which arises when using peer-to-peer software. It is the mandatory installation and set up of third-party software prior to running it. We suggest using the well-known web environment: open your browser and head to the requested service in the web. The new WebRTC standard offers the possibility to establish direct connections between browser instances, thus offering the opportunity of using peer-to-peer techniques in the context of the web. We create a chatting platform based on a peer-to-peer overlay, evaluate its performance and give an overview of possible problems of this new opportunity. In another work, we present an efficient distribution strategy of data that gives guarantees on the distribution time by estimating an upper bound. This method is used in a start-up to realize bandwidth savings in live streaming scenarios using the new standard and peer-to-peer techniques.

We combine the presented methods to the so-called *OverlayMeter*, which provides a basis for extensive monitoring. This basis includes accurate and cost-effective system- and peer-specific data monitoring along with capacity-based peer search. The methods of the OverlayMeter can finally be utilized in various peer-to-peer applications to reliably and precisely monitor the network.

Zusammenfassung

Im letzten Jahrzehnt fanden viele Forschungsaktivitäten im Bereich von Peer-to-Peer-Systemen statt. Applikationen, die das Peer-to-Peer-Paradigma anwenden, sind gegenwärtig und deren Verkehrsaufkommen macht, abhängig von der Region, einen signifikanten Anteil am Gesamtverkehrsaufkommen im Internet aus. Das definierte Ziel der Systeme ist die Schaffung einer gewissen Servicequalität, was in dezentralen Systemen eine Herausforderung darstellt. Diese beruht auf der Tatsache, dass Teilnehmer aufgrund ihrer vorliegenden Informationen Entscheidungen treffen müssen. Um optimale Entscheidungen treffen zu können, ist eine solide und umfangreiche Datenbasis unabdingbar. Die Literatur hält hierfür Monitoringverfahren bereit, worauf in dieser Arbeit aufgebaut wird. Unter Monitoring wird die Datenerfassung und Verteilung von System- und Teilnehmer-spezifischen Daten verstanden. Diese Dissertation beschäftigt sich mit offenen Forschungsfragen zur Verbesserung und Erweiterung von Monitoringverfahren. Ferner werden Fragestellungen zur Vereinfachung von Abläufen zur Inbetriebnahme solcher Peer-to-Peer-Systeme angegangen.

Im ersten Teil beschäftigen wir uns mit Monitoringverfahren im System-spezifischen Kontext. Hier setzen wir auf Baum-basierte Verfahren, denen wir das größte Potenzial zusprechen, und gehen vorhandene Probleme in deren Robustheit an. Modernste Baum-basierte Verfahren in der Literatur werden als hoch präzise aber nicht robust eingestuft, welche der Dynamik der Teilnehmer geschuldet ist. Als Folge werden Kommunikationspfade entlang des Baumes gestört, unterbrochen und im schlimmsten Fall müssen diese rekonstruiert werden. Dies führt zu Datenverlust, was sich negativ auf die Präzision auswirkt. Unsere Antworten auf offene Forschungsfragen umfassen die Einführung von Redundanz in einer smarten Verteilungsfunktion und die Vorstellung zusätzlicher Mechanismen, die für eine verbesserte Wahrnehmung der Monitoringstruktur sorgen. Zwei Verfahren werden im Rahmen dieser Dissertation vorgestellt. Diese erhöhen die Robustheit signifikant, halten die Präzision auf einem gleichbleibend hohem Niveau und weisen ein gutes Kosten-Nutzen-Verhältnis auf.

Neben der Verbesserung von vorhandenen Monitoringsystemen, behandeln wir Phänomene, die zu einer Beeinträchtigung des Monitoringergebnisses führen. Der Einfluss von bösartigen Teilnehmern in Monitoringverfahren wird in der Literatur nicht hinreichend analysiert und vorhandene Lösungen zum allgemeinen Umgang solcher Teilnehmer sind komplex und könnten weitere Angriffsvektoren schaffen. Wir analysieren den Einfluss von bösartigen Teilnehmern, die Ergebnisse manipulieren oder sich nicht an die Protokollspezifikationen halten, betonen die gravierenden Schwachstellen und stellen Mechanismen zur Abschwächung vor. Wir zeigen in einer umfangreichen Evaluation, dass Manipulationsangriffe anhand einer Ausreißererkennung auf eine konvexe Hülle beschränkt und Angriffe auf die Monitoringstruktur durch die Verifizierung der Herkunft gemindert werden können. Insgesamt agiert die Erweiterung passiv und ist somit vielseitig einsetzbar. Zum anderen verfolgen wir ein bisher nicht diskutiertes Feld im Monitoring; der unvollständige Teilnahme von Benutzern in Monitoringverfahren. Wir ermitteln den Einfluss, den das Verhalten auf die Präzision des Monitorings hat, und stellen eine Lösung für das Problem vor. Wir fokussieren uns auf die Einsammlung des globalen Zustands des Systems und sind somit an den Monitoringinformationen aller Benutzer interessiert. Als Lösung stellen wir ein generisches Verfahren vor, welches in Form einer Middleware agiert, um vorhandene Monitoringverfahren wiederverwenden zu können. Hierbei setzen wir auf eine Organisation der aktiven Teilnehmer, um per Probingmechanismen die passiven Teilnehmer zu vermessen und diese Informationen in das Monitoring einzuspeisen. Die Evaluation zeigt eine

hohe Präzision, die unter anderen von der Präzision der Probingverfahren abhängt.

Neben der Erhebung von System-spezifischen Daten gehen wir die Teilnehmer-spezifische Datenerhebung und die Suche nach Knotenkapazitäten an. Das Ziel ist die Schaffung einer effizienten Indizierungsstruktur, welche die vielfältigen und hoch dynamischen Kapazitäten der Teilnehmer effizient und verteilt speichert, und einer präzisen Suche nach Teilnehmern, die spezifische Anforderungen erfüllt. Unsere Motivation für solche Systeme zielt auf die Delegierung von Aufgaben und der Realisierung von verteilten Berechnungen im dezentralen Kontext ab. In der Literatur werden Verfahren vorgestellt, die nicht alle Knotenkapazitäten berücksichtigen, nicht mit der Anzahl an Kapazitäten skalieren oder nicht auf hoch dynamische Daten abzielen. Wir stellen zwei Lösungen vor, die hochdynamische Knotenkapazitäten effizient halten und die Last fair auf alle Teilnehmer verteilen. Während unsere erste Lösung global die Anzahl an Kapazitäten statisch setzt, verwendet unsere zweite Lösung einen dynamischen Ansatz. Die vorgestellten Suchprozesse arbeiten schnell und präzise, womit diese den gestellten Anforderungen genügen.

Im letzten Teil dieser Arbeit richten wir den Fokus auf ein Hemmnis bei der Verwendung von Peer-to-Peer-Software: der Installation und Einrichtung der benötigten Drittanbietersoftware. Wir schlagen die Verwendung von der allseits bekannten Web-Umgebung vor. Benutzer öffnen ihren Browser und nehmen den gewünschten Service in Anspruch, indem die jeweilige Ressource aus dem Web aufgerufen wird. Der neue WebRTC-Standard birgt die Möglichkeit direkte Verbindungen zwischen Browser-Instanzen aufzubauen und bietet somit die Möglichkeit der Verwendung von Peer-to-Peer-Techniken im Kontext des Webs. Wir erschaffen eine Chattingplattform, welche auf ein Peer-to-Peer-Overlay aufbaut, evaluieren die Performance und geben einen Überblick über mögliche Problematiken dieser neuen Möglichkeit. In einer weiteren Arbeit stellen wir eine effiziente Verteilungsstrategie von Daten vor, welche die Verteilungszeit mit einer oberen Schranke abschätzt. Dieses Verfahren wird in einem Start-up verwendet, um mit Hilfe des neuen Standards und Peer-to-Peer-Techniken Bandbreiteneinsparungen in Live-Streaming Szenarien zu realisieren.

Wir fassen die vorgestellten Verfahren zu dem sogenannten *OverlayMeter* zusammen, welche eine Basis für die umfangreiche Erhebung von Daten schafft. Diese Basis umfasst eine präzise und kosteneffiziente System- sowie Teilnehmer-spezifische Datenerhebung und die Suche nach Teilnehmerkapazitäten. Die Verfahren des OverlayMeters können schließlich in diversen Peer-to-Peer-Systemen eingesetzt werden, um dieses zuverlässig und präzise zu vermessen.

Acknowledgements

An dieser Stelle möchte ich mich bei den Menschen bedanken, die mich während meiner Zeit als Doktorand gefördert, inspiriert und unterstützt haben.

Zunächst möchte ich mich herzlich bei meinem Doktorvater Kalman Graffi für all die anregenden, fachlichen Diskussionen und die hervoragende Unterstützung bedanken. Danken möchte ich darüber hinaus meinem Mentor Michael Schöttner für fachlichen Austausch und wertvolles Feedback.

Dem gesamten Lehrstuhl für Rechnernetze und der Arbeitsgruppe für Technik sozialer Netzwerke danke ich für eine sehr interessante Zeit mit einer sehr freundlichen Arbeitsumgebung. Darüber hinaus danke ich Sabine Freese, die immer eine große organisatorische Hilfe war. Besonders danke ich Alexander Schneider und Nhan-Tam Nguyen für deren ausführliches Feedback zu dieser Dissertation.

Recht herzlich möchte ich allen engagierten Studierenden danken, die mich im Rahmen meiner Forschung mit absolvierten Abschlussarbeiten, HiWi-Tätigkeiten und Arbeiten an Veröffentlichungen unterstützt haben. Besonderes danken möchte ich Andreas Funke, Phillip Sandkühler und Christopher Probst für die vielen Diskussionen zu den Arbeiten, welche stets spannend und gleichermaßen produktiv waren. Weiterhin möchte ich Franz Kary, Sebastian Brink und Ivan Tolstun für die sehr gute Arbeit an Prototypen danken.

Zum Schluss möchte ich mich bei meinen Eltern, meinem Bruder und meiner Freundin für den nötigen Rückhalt, die offenen Ohren, die imense Unterstützung, dem freigehaltenen Rücken und dem liebevollen Zuspruch in schwierigen Zeiten danken. Ich kann mich sehr glücklich schätzen Euch zu haben.

Contents

List of Figures

Chapter 1

Introduction

In the last decades the Internet, and in particular its services, have become increasingly important in many areas of everybody's life. The Internet's fundamental basis had been built back in the 1970s with the Advanced Research Projects Agency Network (ARPANET [22]) and has been extended to a world-wide network of networks which went through a commercialization in the 1990s. Back then and nowadays each single person can gain access to this network of tremendous size by signing a contract with an Internet Service Provider (ISP). With the commercialization the number of Internet users skyrocketed. As of the end of June 2017 about 51.8 % of the world's population uses the Internet in their everyday life[1]. Interestingly, in 2016 the number of devices connected to a network, which uses the Internet Protocol (IP), was 2.3 times the global population. Furthermore, Cisco forecasts in their whitepaper[2] that this number will reach 3.5 times the global population by 2021. This is a clear and strong rising trend which will continue at least in the next few years. Another notable trend is that the traffic generated by smartphones will exceed the traffic generated by PCs in a few years[2]. The growth of the number of devices is mostly promoted by mobile and smart devices like smartphones, the Internet of Things (IoT), smart homes and the latest medical gadgets.

Nowadays and in the future, the Internet experiences and will experience changes due to newly arising challenges. At its very beginning the main purpose was to exchange and share research over a relatively wide network. For this principle addressing hosts to obtain or push information, respectively, was a good fit. This particular friendly environment changed with its commercialization. In reaction to this, a very broad discussion about security began and a lot of new protocols were created to protect data transferred over the Internet. Examples are Secure Sockets Layer (SSL) and Transport Layer Security (TLS), asymmetric cryptography proposed by Rivest, Shamir und Adleman (RSA) or Elliptic Curve Cryptography (ECC). Not only has the communication security been changed since then but also the users' sense of privacy. To name an example, nowadays the crypto currency Bitcoin [84] experiences a hype, which uses a decentralized ledger, the blockchain, as its transaction database. In contrast to centralized banking systems a user holding Bitcoins needs to care about and protect its earnings, therefore, strong and reliable security mechanisms are needed and advised to be used.

The aforementioned changes only occur on Open Systems Interconnection (OSI) layers above or below the network layer, where the Internet Protocol is operating, thus, the design principle follows the rule of moving the complexity to the edges of the Internet. In the following we focus

[1] Source: http://www.internetworldstats.com/stats.htm

[2] Cisco VNI hyperconnectivity white paper - source: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html

on layers above the network layer, mainly the application layer, where the majority of changes has taken place. One of the most known protocols in the Internet is the Hypertext Transfer Protocol (HTTP), which spans the well-known World Wide Web consisting of links between hypertext documents. Other popular protocols or services, respectively, are the Domain Name System (DNS) and File Transfer Protocol (FTP). All of them have one in common, they use the client / server paradigm, which means, that the client requests a certain document or resource, which is hosted by a server. Thus, clients utilize services that servers provide and is in the following therefore stated as the *centralized* approach. On the one hand, the centralized paradigm is widely spread in the Internet as it is easy to provide and maintain the service and control the data which is located on the server only. On the other hand there are drawbacks of centralized structures. These include the issue of data availability as the service will be shut down if the server is brought down. More severe implications are the total control of the data by the providers and the accompanying privacy issues. On the contrary, in *decentralized* approaches, such as peer-to-peer (p2p) networks, there does not exist any central controlling instance, thus, each participant implements both functionalities. It provides services to other peers and utilizes services from other peers. As a result, p2p networks need a structure to organize the participating peers, to provide maintenance mechanisms and to offer functionalities to search for resources. They benefit from its decentral structure as these networks are hard to shut down and no instance leeches all data from the users. As mentioned before, the majority of communications over the Internet follows the client / server paradigm and therefore one needs to ask whether it is worth to further analyze the p2p paradigm. In the literature there is research going on how to categorize traffic in the Internet to unveil these shares. Unfortunately, it is hard to classify traffic, as communication is nowadays encrypted and protocols often change the ports used to disguise the running protocol as stated by the authors Dainotti, Pescapè and Claffy [26] and in the work from Iacovazzi and Elovici [59]. Nevertheless, research from 2009 rate the p2p traffic varying between 5 % and 50 % (Pietrzyk, Costeux, Urvoy-Keller and En-Najjary [93] and Maier, Feldmann, Paxson and Allman [78]). Comparing to a small number of applications using p2p networks this is a fairly high share. When evaluating the impact of p2p applications on the generated traffic in the Internet, one needs to take regional differences into account. In China and other countries of the far east territory p2p applications are more popular than for example in Europe. PPStream[3] and PPLive[4] are two famous p2p tv applications providing video content via channels. Each of them builds up a p2p network and video content is distributed amongst the participants. Research was performed for deeper understanding by Hei, Liang, Liang, Liu and Ross [56] where the authors state that the Internet is capable for such p2p tv systems and in future this traffic will even grow. After a short time, in 2008, PPStream proved that and supplied 1.2 million viewers during the live stream of the Olympics in Bejing (2008) as analyzed by Liang, Bi, Wu, Li and Li [73]. Until now, PPStream and PPLive are successful in China and it is still worth mentioning p2p applications in the Internet.

Let us catch a glimpse into the future of the Internet. Firstly, what could happen to the p2p paradigm, if the Internet does not change its protocol layer (IP) and remains on the principle to move the complexity to the edge of the Internet? On the one hand, as mentioned before, p2p networks operating on the application layer will exist alongside with client / server paradigms and may receive increasingly more attention in areas well-fitted for such networks like the streaming of video content or privacy-aware (social) communication networks. On the other hand, newly introduced standards like WebRTC will most probably keep the focus on p2p

[3] Accessible via http://www.ppstream.com
[4] Accessible via http://www.pplive.com

applications. With WebRTC Internet users come into touch with p2p networks when using their browsers to communicate with other users' browsers. Secondly, we refer to the case that the core of the Internet changes. What will happen, if the Internet changes its 'narrow waist', which is the illustration in the literature for the Internet Protocol, and moves from a simple core with all of its intelligence at the edges of the Internet (hosts) to a more complex core? Trends in the literature try to overcome this limitation by introducing fog computing [15], which describes a virtualized platform between devices operating at the edge, and clouds, that are typically spatially distant. Therefore, the fog provides services which are closer to the source or data producer and can thus provide low latency structures. Another approach, which distances itself from the cloud computing paradigm, concentrates mainly on the edge. Edge computing [111] is quite similar to fog computing but does not interact with any cloud services. However it moves computation to the edge instead of pushing data to distant regions in the network to keep the proximity and thus save bandwidth. Software-defined networking [109] is an interesting approach to move the control plane of a network's core to a central instance in order to provide flow management, scalability and flexibility. Hence, till this point we summarize that trends introduce or adapt services to support the data producers, which are located at the edge, and the core of network, which control plane has been reformed. To carry on the change of the Internet's core, a lot of research is going on to change the Internet's host-centric to a content-centric approach in a long term as initially proposed by Jacobson, Smetters, Thornton, Plass, Briggs and Braynard [61]. The vision of content-centric networks is the adaptation of the search behavior from the question 'who stores the desired data?' to 'what data do I desire?' Therefore, this change is not trivial as the Internet's fundamental structure needs to adapt its core to a more complex one. Current research still tackles this fundamental change and its feasibility as discussed by Perino and Varvello [92]. Interestingly enough, p2p plays an important role in this adaptation and therefore may receive a lot more attention in the future. Passarella [89] discusses in its survey reasonable paradigm choices. The author declares Content Delivery Networks (CDNs) and p2p networks to be valuable for such content-centric networks.

During the last 15 years the p2p paradigm, its spanning overlay networks and miscellaneous purposes have been broadly studied in the literature. This research field is mainly motivated by the strong benefits it has as opposed to traditional client / server paradigms. Probably the most important point from the users' point of view is the distributed data storage and its implication on the users' privacy. The latter is protected by the nature of having no central index and, as a consequence, a significant increased complexity of tracking the consume behavior of a specific user. From the system provider's point of view there are a lot of advantages as it is relatively hard to shutdown this system in contrast to the centralized approach, where it is sufficient to attack, overload, eclipse or ultimately take over the central unit. Furthermore, the system provider does not need to build up expensive server farms as the participating peers contribute to the system with their bandwidth to help to maintain and run the system. As a result, the operational costs for provider are near zero even if the system expands and grows to a big one. Hence, we attribute a superior scalability mainly because of the absence of a central unit, which would be inquired by each participant, and the peers' contribution of their (upload) bandwidth to the system. In fact, there are a lot of devices at the edge of the Internet with unutilized capacities. This is the point, where the motivation of p2p and edge computing comes into play. These devices at the edge can be used to offer services to each other. The services face algorithmic challenges as they call the need of routing mechanisms in decentralized systems with a high number of devices. For example, given a search query for a particular object, which is maintained by the system, it needs to find a route to the object by either taking distributed

indexing structures or intelligent query propagation into account. In general, we mention it as a problem due to the decentralization of a distributed system, which needs the cooperation of all participants to provide a certain structure in a self-adaptive and self-organizing manner and to implement mechanisms to offer and claim services to its participants.

However, as each node in a distributed system must locally decide to which node it connects, it forwards requests to or replicates data at, these system mechanisms' solutions always underlie a distributed optimization problem. It is of high importance to solve such problems to enhance the system's experience. We state that the more information each node has, the better they can optimize their decisions. This demand for information can be met by *decentralized monitoring solutions*. Such solutions collect data about the system's state, like the number of hops needed to answer a request, the time needed to find the requested resource and the number of replications per resource. Moreover, such systems can monitor capabilities of the system's participants and provide an infrastructure to find nodes fulfilling certain capabilities. Without using any central instance, which gathers and disseminates monitoring information, this is a challenging task as the monitoring data is scattered across all nodes. Monitoring solutions therefore solve a decentralized distribution problem. In a nutshell, the main goal of a monitoring solution is to gather information on the system's state and its participants and provide a structure to query for information or disseminate it to its users, if required. To provide these information to the nodes, it is of interest to introduce and optimize monitoring solutions. Therefore, we tackle in this dissertation open questions in the field of decentralized monitoring to improve the experience of decentralized p2p networks and provide new features to them.

In particular, we focus on the following challenges, which are picked up and are further detailed in the next section. One of the most challenging side effects in decentralized systems is owed to the dynamism of peers. As peers are subject to churn, which is the procedure of participants going on- and offline at their own will, structures used for gathering and disseminating monitoring information get disturbed which leads to a loss of information. It is a demanding task to find the sweet spot between the criteria of a high accuracy, robustness and efficiency in a well-scaling solution. Furthermore, the investigation of maliciously behaving nodes and their influence in a cooperative environment, like the decentralized monitoring, is important. It needs to study whether various malicious behavior patterns significantly influence the monitoring solution's outcome and how it can be mitigated. In addition, nodes might not be aware of the monitoring solution running in parallel to the p2p application. Such a case can easily happen in a network with independent and distributed peers running different versions of the protocol. Generally, monitoring protocols do not assume such a case but it has a greater impact on the accuracy and it needs mechanisms to overcome this issue. Besides the continuous monitoring of the decentralized system's state, it is of interest to provide a capacity-based peer search. To be able to search for peers fulfilling certain capabilities introduces new freedom to edge computing and future applications. Lastly, we aim to speed up the applicability of p2p solutions by using the latest web standard WebRTC. We investigate the feasibility of browsers running a p2p network, discuss arising problems and explain emerging opportunities when porting p2p software to the web.

In the following Section 1.1, we detail the functional requirements for decentralized monitoring systems and provide definitions on the problem statements we just outlined. Further, we formulate our research questions in Section 1.2 and summarize our contributions in Section 1.3. Lastly, we give an outline of this thesis in Section 1.4.

1.1 Motivation and Problem Statements

After having motivated monitoring in decentralized environments and outlined arising challenges, we detail these challenges in this section. Prior to this, we first give a definition and a functional description on monitoring in general and then refer to monitoring classes we focus on. We define monitoring in a decentralized communication network as a service of which the goal is to gather monitoring information from its participants, create an evaluated view based on the information basis and offer mechanisms to obtain this view. The first step, the gathering of the scattered information, is performed either in a continuous fashion or on demand. In this thesis we focus on system-wide continuous information gathering, that means that the process targets to involve each participant's view and the resulting evaluated view is always kept up-to-date. The resulting view is either pushed to participants or the peers need to pull the information from the service. Depending on the gathered type of information, we either focus on the push-based or pull-based approach. The first type, namely the system-specific information, together with challenges that arise is thematized next. Afterwards, we refer to the peer-specific information monitoring.

Monitoring system-specific information stands for gathering and disseminating information about the system's state. Examples for metrics are the number of hops and time needed for the routing mechanism or the number of replications for an object. In the context of monitoring system-specific information, the evaluated view is mentioned as the global view and it is disseminated using a push-based approach. For efficient gathering and dissemination of data, it is of high interest to aggregate data to run a scalable approach saving storage space and therefore bandwidth. Aggregating data means to combine several data objects to only one object with certain attributes according to aggregation functions. These aggregation functions are for example the minimum, maximum, mean, sum, standard deviation etc. of the data combined to one object. Fortunately, system-specific information is predestined to be aggregated. To give an example, imagine two data sets d_1 and d_2 which hold information of the monitoring metric hop count with the aggregatable attributes min, max, mean, count and sum. Let $d_1 = [\min = 1, \max = 5, \text{mean} = 3, \text{count} = 2, \text{sum} = 6]$ and $d_2 = [\min = 2, \max = 7, \text{mean} = 4.3, \text{count} = 3, \text{sum} = 13]$. These data sets can be aggregated to the object $d_{\text{aggregated}} = [\min = 1, \max = 7, \text{mean} = 3.8, \text{count} = 5, \text{sum} = 19]$. Therefore, by calculating aggregation functions and combining data objects, the amount of data sent during the gathering and dissemination process can be dramatically decreased. Lastly, we refer to the question on how to gather and disseminate aggregated data. The literature in the field of decentralized monitoring solutions, also stated as decentralized aggregation schemes, offers three different main concepts for the structure (cf. surveys from Makhloufi, Doyen, Guillaume, Bonnet, and Gaïti [79] and Jesus, Baquero and Almeida [66]). The process follows either a tree-based (structured), gossip-based (unstructured) or hybrid approach. In the former its participants follow a responsibility pattern, a tree, to collect and disseminate monitoring information by aggregating it. The gossip-based approach does not dictate any responsibility pattern, instead, peers gossip with each other to approximate their view to the global view. Lastly, hybrid approaches combine the two approaches, thus building up an aggregating tree and introducing gossip communications. Gossip-based or hybrid approaches can not achieve high accuracy as they implement an estimation of data. Furthermore the former can not become efficient as no defined task sharing is implemented. In this thesis we focus on tree-based approaches as they provide a highly scalable tree to guarantee high accuracy and efficient data transport and therefore we attribute the greatest potential to it. Using a tree to monitor system-specific information has three challenges, which we deal with in this thesis.

The first challenge, which is defined as Problem Statement PS1, concerns the robustness of the tree structure when perceiving node dynamism. As the tree structure depends on peer responsibilities, which change on churn events, a reconstruction of the tree is enforced. As a result, this leads to data loss and a degradation of the accuracy due to a reduction of robustness. Generally speaking, mechanisms are needed to mitigate the devaluation of the tree's robustness. The following two challenges are mentioned as restrictions that impair the main goal of (tree-based) monitoring solutions, that is, efficiently providing accurate monitoring information. The second challenge, in the following mentioned as Problem Statement PS2, tackles malicious behavior in the tree-based monitoring environment. Imagine a perfect working monitoring protocol, in terms of its accuracy, robustness and costs, is going online and participants are able to monitor system-specific and peer-related information. As long as all participants behave honestly and forward data according to the protocol specifications, there may be no concerns about it. But the history of the Internet for example has shown that groups of people will always try to challenge a system's security. Thus, they may try to challenge the system by manipulating data or circumventing policies or protocols' rules. Hence, an unprotected but very well performing monitoring mechanism providing a high accuracy is doomed to fail if maliciously behaving peers enter it. By manipulating data or not complying with the protocol specifications, the global view obtained from the monitoring solution may get influenced which might lead to erroneous decisions for applications using this view. Therefore, it needs mechanisms to mitigate the influence malicious behavior has on the accuracy of the monitoring solution. Lastly, the third and last challenge in the field of monitoring system-specific information with a tree-based solution is referred as Problem Statement PS3. It deals with the particular problem of the incomplete participation of peers in monitoring solutions. This means that only a part of all participants of the p2p network also participates in the monitoring mechanism, as they might do not know about the existence of such. This can happen when having a look on the update behavior in p2p applications. As users are responsible for updating their application, they might not or not yet install an update containing an important feature. This feature can be a monitoring mechanism, therefore, during a period of time the monitoring mechanism does not contain the data from those peers who did not yet update their software. The main problem occurs when having a look on state-of-the-art system-wide monitoring solutions which do not offer an adequate handling and therefore the peers continue to assume that all peers of the p2p network also participate in the monitoring mechanism. Instruments are needed to overcome this issue and mitigate any reasons which lead to a decrease in the monitoring's accuracy.

Now, we turn the attention to peer-specific information monitoring. In the following, we define how data looks like, what the differences to system-specific monitoring are, which use-cases we propose and what the challenge in this field is. Peer-specific information is defined as capacities of peers available for utilization, for example bandwidth for up- and downstream, persistent storage, CPU cycles and transient storage (RAM). For example, a certain peer A could hold the following capacities: (CPU = 1500 MHz; UploadBandwidth = 100 KBit/s; RAM = 500 MB; Storage = 2 GByte). The evaluated view of the peer-specific monitoring is an actual mapping of the peer to its capacity. Therefore, this information is not aggregatable as the mapping between the capacity and the peer providing this capacity must be kept. Furthermore, gathering and disseminating data in a tree-based manner, which would be the analogue to tree-based system-specific monitoring, would result in a long not aggregatable map, thus, this naive approach would not scale. Instead, in the literature the monitoring solution SkyEye.KOM [51] trims this map on each level in the tree to keep the load scalable but the search's outcome does not consider all participants. Alternative approaches are multidimensional indexing schemes

as presented in the survey from Zhang, Xiao, Tang and Tang [134]. Unfortunately, these are not suitable to solve our problem statement as these schemes are performant in accessing static data like meta data from documents and are therefore not optimized for highly dynamic data like peer capacities. Moreover, peer capacities are expected to be highly dynamic, thus they may change frequently. Peers may obtain (parts of the) evaluated view in a pull-based manner, which we refer to as the *capacity-based search*. So, the gathered peer-specific data must be provided by the system and peers can utilize the capacity-based search to query for peers with certain capacities. By utilizing the capacity-based search, one can imaging future applications for distributed computing, supporting overloaded peers or job delegation in general. In the following we specify such use-cases in more detail. Distributed computing is an appealing scenario in a pure p2p network. Here, we choose a push-based approach, which means that a task is created and pushed to an appropriate peer which processes it. Peers' capacities are highly heterogeneous and therefore a mechanism to search for suitable peers and their capacities, respectively, is needed. Eventually, after a set of suitable peers has been found, tasks are forwarded to them and these peers return the outcome, if any, to the requester. Another use-case for being able to search for peers with certain capacities is the support for overloaded peers, as proposed by Graffi, Groß, Stingl, Nguyen, Kovacevic and Steinmetz [51]. Heterogeneous peers in homogeneous p2p networks might perceive a status of being overloaded. The status of being on high load means that the peer needs to spend all of its capacities to react on incoming requests. A peer is said to be overloaded if it can not process all requests in-time and therefore requests need to be queued in a waiting queue. This harms the overall quality of service of the application if peers are not able to handle requests in-time, or even if peers are able to process them in-time, the quality of service might aim for a minimum transfer rate which can not be held. In that case, an overloaded peer could search for suitable peers to support them by processing requests in-time and provide a certain transfer rate to hold the quality of service. The main challenges in this field are as follows. Firstly, due to the dynamism of peer-specific information and the fact that the gathering process can not aggregate the data, it needs a dedicated structure to process and store the evaluated view. Furthermore, the evaluated view will be queried by peers to search for peers with certain capacities. A more detailed definition and description is stated in Problem Statement PS4.

Our last challenge we tackle in this thesis concerns the fundamental of monitoring systems, the p2p networks. We state that current p2p applications are too complex to access - third-party software needs to be installed and configured. This scares off (unexperienced) users to try out new software. We aim to speed up the process to get in touch with new software by using the browser. With the help of WebRTC browser-based p2p networks can be set up. In Problem Statement PS5 we aim for a proof of work and investigate potential performance and technological weaknesses.

In the following we detail the five problem statements we motivated and outlined in this section. The succeeding paragraphs have a fixed structure. First, we give a formal definition of the problem statement by specifying what is given, which goals we have and which optimization problems must be solved while considering particular constraints we care about. Afterwards, we give a verbal description of the problem statement.

PS1: Increasing the Robustness in System-specific and Tree-based Monitoring Systems

(Given)

1. Each node $n \in \mathcal{N}$ holds a continuous local system-specific information pool for the metrics $\mathcal{M} = \{m_1, \ldots, m_d\}$. The local monitoring data at time t is defined as the d-tuple

   $lmd_n^t := \langle v_{n_1}, \ldots, v_{n_d} \rangle$ with $v_{n_i} \in \mathbb{R}$ being the value from n for metric m_i

2. From the global point of view the following sets are given defining responsibilities in the tree-based monitoring structure:

   $$\forall\, n \in \mathcal{N} : \exists\, \mathcal{N}_{Sub_n} \subseteq \mathcal{N} \text{ with } n \in \mathcal{N}_{Sub_n}$$

   Therefore the subset $\mathcal{N}_{Sub_n}$ contains n and the nodes that are directly subordinate to n and are thus the sub-tree rooted at n.

(Goal)

$$\exists!\, n \in \mathcal{N} : \mathcal{N}_{Sub_n} = \mathcal{N}, gmd^t := amd_n^t$$

$$\text{whereby, } amd_n^t := \text{aggr}(lmd_n^t, amd_{k_1}^t, \ldots, amd_{k_i}^t)$$

$$\text{with pairwise distinct } k_1, \ldots, k_i \in \mathcal{N}_{Sub_n} \setminus \{n\}, i = |\mathcal{N}_{Sub_n}| - 1$$

The goal is to obtain the global monitoring data gmd which is calculated at the root node. For this, it needs the recursively defined amd_n^t aggregated monitoring data, which is defined as the aggregation of the local monitoring data from n and the aggregated monitoring data from all its children (all $k \in \mathcal{N}_{Sub_n} \setminus \{n\}$) in the tree structure. The aggregation function aggr takes either local monitoring data or aggregated monitoring data as an argument and calculates the aggregate using all aggregate functions $\mathcal{F}$. It contains aggregate functions like the minimum, maximum, sum, mean, variance and standard deviation.

(Optimization Problem 1) *Decentralization*: Approximate the global view for any t and minimize the estimation error:

$$\text{Let } gmd^{t'} \text{ be the approximation at time } t$$

$$gmd^{t'} = gmd^t + \epsilon, \text{ with } |\epsilon| \text{ minimal}$$

(Optimization Problem 2) *Distribution*: The global view $gmd^{t'}$ must be made available for and pushed to any node $n \in \mathcal{N}$. The delay should be minimal.

(Constraints) Nodes $n \in \mathcal{N}$ can go on- and offline occasionally. Therefore the set $\mathcal{N}$ is subject to change.

For the first problem statement, two elements are given. The first element is a sensor each participant maintains which can be polled for the current local system-specific monitoring data. The second element defines the responsibility structure of the tree-based monitoring approach from a global point of view. In our definition it is specified in a top-down manner, hence, the peer n is responsible for the sub-tree induced by it. The goal of the system-specific

monitoring is to obtain the global view containing all local views of the participants by using the gathering and dissemination processes of the monitoring approach. State-of-the-art tree-based monitoring approaches gather the information in a bottom-up manner and disseminate the global view in a top-down manner. Therefore, the root node is the one performing the last aggregation and holding the global view. We specify two optimization problems in the gathering and dissemination processes. Firstly, the distributed calculated global view obtained by the tree structure should equal the actual global view, therefore, the error between them should be minimal to maximize its accuracy. Secondly, the global view must be made available with minimal delay to each participant in the monitoring approach. The constraint for this problem statement is as follows. Participants are subject to churn, therefore they can go on- and offline occasionally. We define that going offline is equivalent to pulling the power plug, therefore, there are no notifications sent out to neighboring peers or similar. State-of-the-art monitoring systems like SkyEye.KOM [52] offer a high accuracy and adaptivity with a high efficiency but perform poorly under the constraint of churn. This is due to the tree-based structure which is prone to churn and is mentioned to be not robust. We give a simplified example showing the problems tree structures have with churn in the following. Figure 1.1 shows in the upper half a scenario, where the participants of a tree-based monitoring system gather aggregated views and send them towards the root node. However, when a node leaves the structure, the responsibilities change and the tree must reconstruct itself. Nodes arriving at new positions must clear their local data base for monitoring data to not introduce duplicates to the system. Amongst other problems, these nodes continue to send data upwards the tree. This data does not contain data from their potential new children nodes as they did not recognize their position yet. Eventually, the children nodes distribute their data to the (new) parent in their next step. Thus, the reconstructing of the tree leads to data loss across the whole tree and therefore degrades the global view's accuracy. We conclude that both optimization problems are solved by state-of-the-art tree-based monitoring approaches. However, with the constraint taken into account, which is the phenomenon of churn, the accuracy collapses. So, in a real world scenario we need a high accuracy approach, which is robust and efficient in terms of costs, as we can not assume that nodes do not depart from or arrive by the system.

PS2: Malicious Behavior in System-specific and Tree-based Monitoring Systems

(Given) See PS1.

(Goal) See PS1.

(Constraints)

1. Segmentation of all nodes into the groups of honest and malicious nodes.

$$\mathcal{N}_{\text{honest}} := \{n | n \in \mathcal{N} \wedge \text{ n follows protocol specifications}\}$$

$$\mathcal{N}_{\text{malicious}} := \{n | n \in \mathcal{N} \wedge \text{ n behaves maliciously}\}$$

$$\mathcal{N} = \mathcal{N}_{\text{malicious}} \cup \mathcal{N}_{\text{honest}} \text{ , } \mathcal{N}_{\text{malicious}} \cap \mathcal{N}_{\text{honest}} = \emptyset$$

2. Malicious nodes $n \in \mathcal{N}_{\text{malicious}}$ manipulate their aggregated monitoring data amd_n^t at their will or do not follow protocol specifications.

The second problem statement is concerning the malicious behavior of participants in monitoring mechanisms. The given and goal sections refer to the corresponding sections of Problem

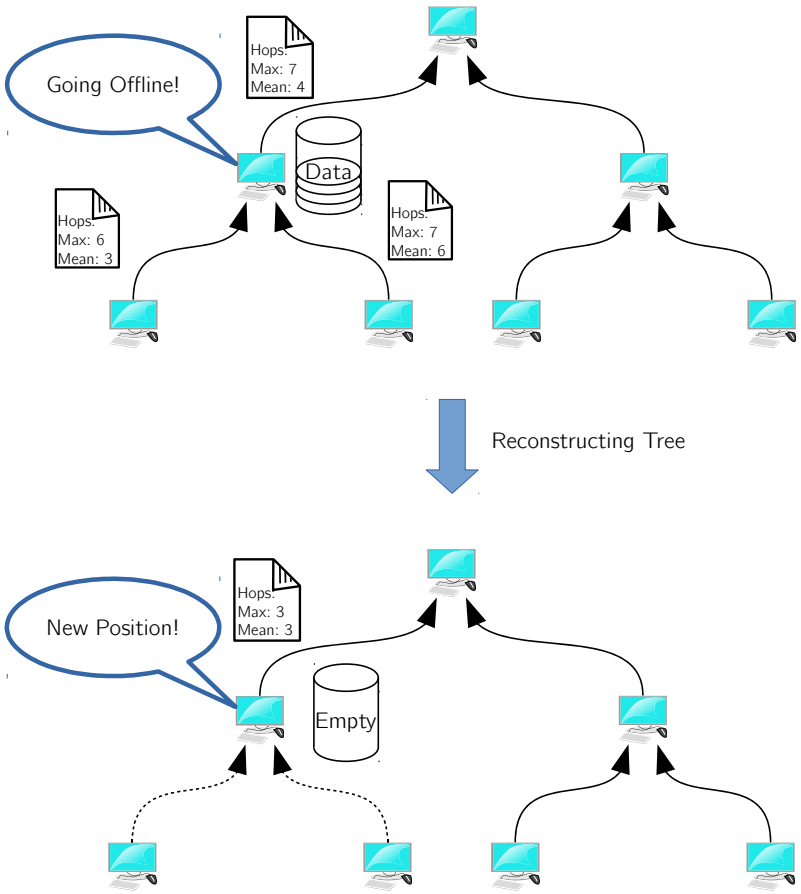


Figure 1.1: Simplified scenario where one node departs from the monitoring tree. While the tree is reconstructing, the node, which takes the new position, needs to clear its monitoring data base. As its children are still not available, it starts from scratch and propagates only its local monitoring data up the tree. Eventually, its new children send their data to the new node in their next distribution step.

Statement PS1 and therefore a system-specific monitoring approach is operating which aims for a precise global view. Only the constraint has changed, which is, the presence of malicious behavior. In detail, all participants are segmented into honest and malicious nodes, as specified in the first constraint. The former group contributes to the system and follows protocol specifications, whereby the latter behaves maliciously in any way. The second constraint indicates the malicious behavior of nodes which is the manipulation of their aggregated view or the non-compliance with protocol specifications. An example of nodes behaving maliciously by manipulating data is visualized in Figure 1.2, in which a scenario with a single malicious peer in a tree-based monitoring system is shown. The intruder is located at a higher level in the tree and can therefore manipulate the data representing the whole sub-tree. Its parent does not verify incoming data per default, therefore this data is accepted by the system. Hence, an analysis of the impact of malicious behavior and the development of its countermeasures are crucial. Furthermore, the motivation of malicious behavior is versatile and needs to be

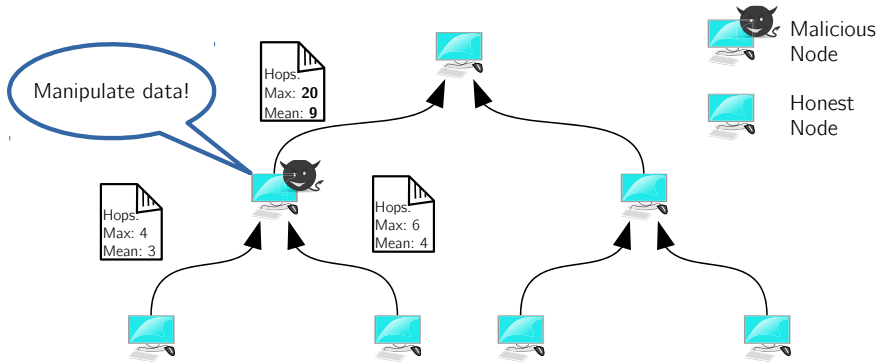


Figure 1.2: Scenario with one malicious node in a tree-based monitoring system. The node regularly follows the monitoring protocol but manipulates the data and sends this data up to the root, which accepts the data.

specified. The literature provides general solutions introducing systems with trust mechanisms like the one presented in EigenTrust [67], which are commonly accepted and used in p2p environments. However, it does not provide any mechanisms tailored for specific domains of monitoring approaches.

PS3: Partial Participation in System-specific and Tree-based Monitoring Systems

(Given) See PS1.

(Goal) See PS1.

(Constraints) In addition to constraints from PS1:

1. Segmentation of all nodes into the groups of passive and active nodes.

$$\mathcal{N}_{\text{active}} := \{n | n \in \mathcal{N} \wedge \text{ n follows protocol specifications}\}$$

$$\mathcal{N}_{\text{passive}} := \{n | n \in \mathcal{N} \wedge \text{ n behaves passively}\}$$

$$\mathcal{N} = \mathcal{N}_{\text{passive}} \cup \mathcal{N}_{\text{active}} \, , \, \mathcal{N}_{\text{passive}} \cap \mathcal{N}_{\text{active}} = \emptyset$$

2. $\forall n \in \mathcal{N}_{\text{passive}} : amd_n^t = \emptyset$.

The third problem statement deals with partial participation in the focused class of monitoring systems. Like in the second problem statement, we inherit the given and goal sections from the first problem statement and therefore aim for a precise system-specific monitoring system. In addition to the constraints in the first problem statement, we consider a segmentation of all participants. Thus, the first constraint shows that nodes from N_{active} are mentioned as nodes which actively participate on the monitoring system, whereas nodes from $N_{passive}$ are not participating. The second constraint says that passive nodes actually empty their aggregate view by practically dropping the information as they are not aware of the system. When having a glimpse on the goal formula, this behavior has a crucial impact on the calculation of the global view. Figure 1.3 visualizes this statement. In the given scenario the topology

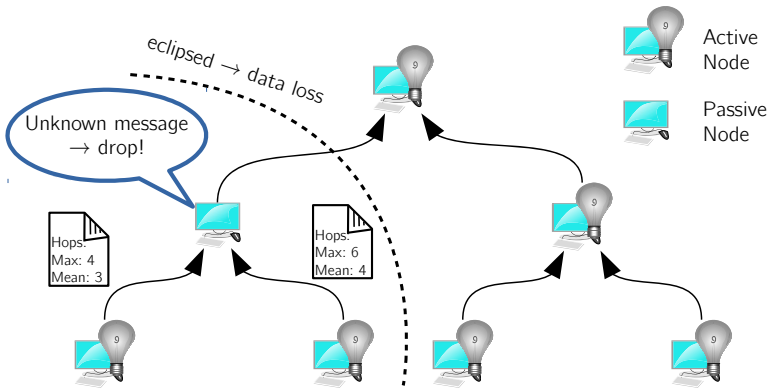


Figure 1.3: Scenario of a tree-based monitoring system with six active and one passive node. The latter receives monitoring data from its children, as they assume it participates in the system. The passive node can not handle this message type and therefore drops this message, which inevitably leads to an eclipsed sub-tree and data loss.

of a tree-based monitoring system is outlined. Six out of seven peers are active ones, which are aware of the running monitoring approach, and therefore exchange monitoring information within the topology. However, the passive peer, which does not implement the monitoring mechanism, will drop incoming monitoring messages, which are unknown to it. As this peer has a position as an inner node, this message drop will unavoidably lead to an eclipsed sub-tree and data loss in a large-scale. We need to note that nodes from $N_{passive}$ are not behaving maliciously, therefore they do not drop data intentionally. We state that these nodes do not refuse any communication per se, instead they are willing to cooperate indirectly. Therefore, we need mechanisms to provide a precise monitoring for both active and passive peers.

PS4: Peer-specific Monitoring in Peer-to-Peer Networks

(Given)

1. Each $n \in \mathcal{N}$ holds a continuous local peer-specific information pool for the peer capacities $\mathcal{C} = \{c_1, \ldots, c_d\}$. The local monitoring data at time t is defined as the d-tuple

   $lmd_n^t := \langle v_{n_1}, \ldots, v_{n_d} \rangle$ with $v_{n_i} \in \mathbb{R}$ being the value from n for capacity c_i

2. K- or full-search queries in the form

$$q = \left\langle [v_{1_{\text{low}}}; v_{1_{\text{high}}}], \ldots, [v_{d_{\text{low}}}; v_{d_{\text{high}}}] \right\rangle$$

(Goal)

1. Create a structure that indexes lmd_n^t for all $n \in \mathcal{N}$:

$$\phi : \mathcal{N} \to \mathbb{R} \times \cdots \times \mathbb{R}, n \mapsto lmd_n^t$$

2. Provide a mechanism for capacity-based peer search, which is the retrieval of peers fulfilling the requirements of the query q:

$$q_{\text{result}} := \{n | n \in \mathcal{N} \wedge lmd_n^t \text{ matches } q\}$$

$$lmd_n^t \text{ matches } q \Leftrightarrow v_{n_1} \in [v_{1_{\text{low}}}; v_{1_{\text{high}}}] \wedge \cdots \wedge v_{n_d} \in [v_{d_{\text{low}}}; v_{d_{\text{high}}}]$$

(Optimization Problem 1) *Decentralization*: Approximate the optimal indexing scheme ϕ in a decentralized environment for any t:

$$\phi' : \mathcal{N} \to \mathbb{R} \times \cdots \times \mathbb{R}, n \mapsto lmd_n^t + \epsilon, \text{ with } |\epsilon| \text{ minimal}$$

(Optimization Problem 2) *Search*: Provide a high accuracy capacity-based search by querying data from the indexing scheme. The obtained result q'_{result} from ϕ' should equal the optimal q_{result} from ϕ.

$$\text{optimal recall: } \frac{|q'_{\text{result}} \cap q_{\text{result}}|}{|q_{\text{result}}|} \to 1$$

$$\text{optimal false positives: } \frac{|q'_{\text{result}} \setminus q_{\text{result}}|}{|q_{\text{result}}|} \to 0$$

(Constraints) The constraints are mentioned as *variants* for potential solutions.

1. Variant 1: fixed number of capacities and granularity
   a) $|\mathcal{C}| = d$ is fixed, therefore no new capacities can be added during runtime.
   b) $\forall c \in \mathcal{C} : v_i \in \mathbb{N}|_{[0;f_{c_i}]}, \forall i, f_{c_i} > 0$ and fixed
2. Variant 2: dynamic number of capacities and granularity
   a) $\mathcal{C}$ is unbounded and d mutable, therefore new capacities can be added during runtime.
   b) $\forall c \in \mathcal{C} : v_i \in \mathbb{R}, \forall i$

Our fourth problem statement deals with the monitoring of peer-specific information. We indicate having two elements given. The first is the local peer-specific information a peer holds. We define it similar to the system-specific information except that the set $\mathcal{C}$ is different. It contains peer-related capacities like the available storage space or bandwidth. In general, peer-specific information are highly dynamic as they represent capabilities of devices the software runs on and can therefore be exposed to frequent changes. The second element given is a search query which is needed for the capacity-based search. It consists of d intervals, one for each capacity, whereby each interval specifies the search range for the corresponding capacity. Here, we introduce two search methods: the k- and full-search. The former is used to get k hits in the specified query range, whereas the latter responds with all peers in the range. The goal is defined as follows and is sketched in Figure 1.4. Firstly, it needs an indexing structure due to the fact that capacities are not aggregatable as stated beforehand. Therefore, a mapping from peers to their capacities is needed and is mentioned as indexing scheme ϕ. Secondly, a mechanism providing the capacity-based search is needed. Participants must be able to query the indexing structure or the inverse function ϕ^{-1}, respectively, to obtain all peers that meet the requirements of the query. Requirements are met, if and only if a peer's capacity value for each capacity is within the interval specified by the query. We elaborate this in a small example. Imagine a small network of three nodes N_1, N_2 and N_3. Let $\mathcal{C} = \{$ RAM, CPU $\}$. For an arbitrary t, the following applies: $lmd_{N_1}^t = \langle 100, 500 \rangle$, $lmd_{N_2}^t = \langle 500, 400 \rangle$ and $lmd_{N_3}^t = \langle 1000, 900 \rangle$. Let $q = \langle [400; 900], [100; 600] \rangle$ be a full-search query. Then, the result of the search query is as follows $q_{\text{result}} = \{N_2\}$, as N_2 is the only peer fulfilling both requirements. Back to the definition of our goals, we declare the following optimization problems to reach the goals. The first optimizing problem concerns the approximation of the optimal indexing scheme in

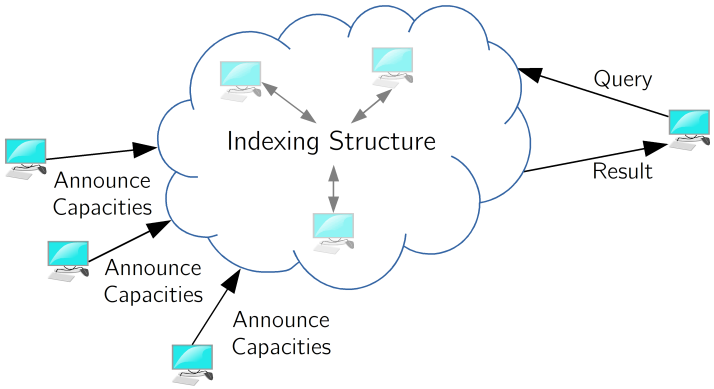


Figure 1.4: Overview of the indexing of capacities and the capacity-based search. Peers announce their capacities to the indexing structure. The indexing structure efficiently stores the capacities and provides a mechanism to search for them. Peers may query for capacities and get the result from the indexing structure.

a decentralized environment. Therefore we state, that the error induced by the maintenance of the decentral environment should be minimal. The second optimizing problem refers to the capacity-based search and its performance. We aim to reach a high recall, thus we are able to obtain all peers in the queried space, and we aim for a low false positive rate, that is zero in the optimum, while keeping the efficiency of the query process high. Lastly, we specify constrains of this problem statement and define two variants for the indexing structure. The first variant calls for a fixed number of capacities and for each capacity a fixed number of classes. This means that each capacity, for example the upload bandwidth, must be divided into disjoint ranges (classes), for example $0 - 50$ KBit/s, $50 - 1000$ KBit/s and 1 MBit/s $- \infty$. Whereas the number of capacities and number of classes per capacity is dynamic and unbounded for the other variant. In summary, we aim for an accurate indexing scheme for highly dynamic data providing a precise capacity-based search considering all peers' data in the required range.

PS5: Speeding up the Applicability of Peer-to-Peer Networks In the fifth and last problem statement we deal with a problem of the monitoring system's basis. This part contains a functional description on the problem statement without any formal definitions. We state that, entering a p2p system to access a certain service is much more complex than accessing a service hosted on a web server. Significant additional steps are required because a p2p application needs to be installed, configured properly, firewall ports may need to be opened etc. We think that this is a huge discouragement to enter, test and use such systems. Internet users in general are used to web applications - firing up the browser and heading directly to the needed service. Therefore, a technology is needed which abstracts all of these barriers. Fortunately, Web Real-Time Communication (WebRTC) is being standardized which, in short, allows a p2p connection between two browser instances. Hence, it eliminates the installation and configuration process as latest prominent browsers support WebRTC. By fast accessing a p2p service via the browser we see a prospect of attracting more potential users. Our main goal is to elaborate whether the new standard is suitable to run (pure) p2p networks with browsers. A proof of work is required to analyze the feasibility of p2p networks in such limited environments. The web platform includes restrictions which may lead to performance problems and other constraints it needs to investigate.

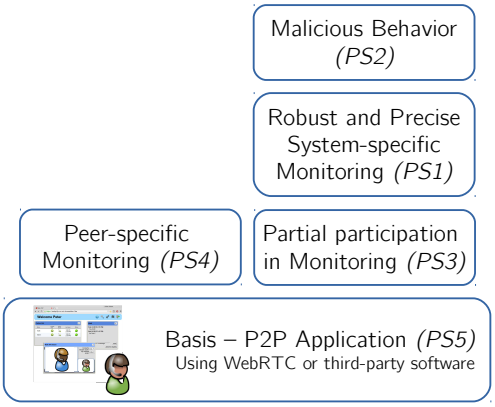


Figure 1.5: Big picture of the problem statements. PS5 deals with the way an end-user interacts with p2p software, whereas the Problem Statements PS1 – PS4 refer to the monitoring services. PS3 is the fundamental basis for PS1. The security is built atop the system-specific monitoring and tackles malicious behavior in PS2.

At the end of this section we recap and interconnect our problem statements and refer therefore to Figure 1.5. First, by tackling Problem Statement PS5, we start with a p2p application running either in the web environment using WebRTC or in a third-party software. This can be either an already running p2p solution or a newly developed. Building on this basis we offer either the stack for peer-specific or system-specific monitoring. The former is solved in Problem Statement PS4, which provides an accurate and fast adapting indexing scheme suitable for highly dynamic peer-specific information and yields a capacity-based search mechanism. For the latter we rely on a tree-based monitoring structure to gather all local views and disseminate the resulting global view. Prior to operating this monitoring feature, we solve Problem Statement PS3 and therefore introduce a mechanism to include a monitoring mechanism in a running environment, for example per software update. Building on it, by offering robust and precise gathering and disseminating processes, we are able to tackle the Problem Statement PS1. Furthermore, challenging PS2 will mitigate malicious behavior in the solution and therefore prevents from distorting the precise monitoring view. By solving Problem Statements PS1 – PS5 we achieve an *OverlayMeter*, the enhancement of state-of-the-art monitoring of system-specific information and peer-related capacities in p2p networks.

1.2 Research Questions

In this section we present and discuss research questions according to the Problem Statements PS1 – PS5 from Section 1.1. These questions are the central ones during our research. In the following we refer to one problem statement after the other and we pursue a fixed structure within them. In each problem statement paragraph we first shortly recap the objective and then lead over to questions in three categories. At first, we specify questions about the design or protocol specifications, respectively. Then, we indicate questions about the solution's performance and its efficiency.

PS1: Increasing the Robustness in System-specific and Tree-based Monitoring Systems In the first problem statement we refer to the lack of robustness in state-of-the-art tree-based monitoring solutions. We concentrate on tree-based monitoring approaches as we attribute greatest potential to them. The lack of robustness is due to the fragile structure and its proneness to churn. Hence, we aim for an approach enhancing the robustness and thus keep the accuracy high, even in scenarios with high node fluctuation.

PS1.1 How can the robustness of tree-based monitoring systems be enhanced by keeping the exact precision? In particular, how can a tree-based communication structure be improved to provide a certain robustness in times of continuous reconstructing communication paths? These questions are fundamental for the orientation of the solution and were therefore our guiding questions.

Once answered the protocol related question we measure the solution's quality by respecting the following questions:

PS1.2 What is the enhancement factor and how is the precision of this approach in a dynamic environment? We need to measure the enhancement and its impact on both, the precision and robustness. Furthermore, it is of high interest to investigate scenarios in an increasingly dynamic environment where peers join and leave the network at their own will.

PS1.3 What is the price of the increased robustness? Most likely the particular enhancement will introduce additional costs, which must be considered for the evaluation.

PS1.4 Is the increased robustness worth the increased costs, and furthermore, what is the cost-benefit ratio? Lastly, the improvements and the costs must be carefully considered. It must be evaluated whether the trade-off is an actual overall improvement by having a closer look on the cost-benefit ratio.

PS2: Malicious Behavior in System-specific and Tree-based Monitoring Systems The second problem statement concerns the security of tree-based monitoring solutions with its main motivation lying in severe consequences when using its potentially manipulated outcome. Therefore the objective is to mitigate the impact malicious behavior has on the accuracy of the system. We concentrate on the following research questions which address the need of a security model and initial steps for its design.

PS2.1 What attacks are possible, what is the motivation of those attackers and how do they influence the monitoring's outcome? These questions are very central ones as further analysis on attack vectors is essential to understand the security problems of monitoring solutions. Therefore, first attack vectors give a proof of concept. Besides, the influence on the global view provided by the particular monitoring solution is crucial and should be carefully inspected. This leads us to the next question.

PS2.2 How does malicious behavior affect these approaches? This question has to be answered in order to see the extent of the influence of malicious behavior. Furthermore, by understanding the impairment, countermeasures can be designed, what in turn leads us to the next question.

PS2.3 How can the malicious behavior be mitigated? This short question is a far-reaching one. The just presented malicious behavior needs to get mitigated and this research question handles the 'how' of it. Therefore, it is part of the solution and the most important one in this problem statement. We concentrate on the mitigation of malicious behavior as a complete suspend of such behavior is most probably not achievable due to the complexity of its nature.

In the following we refer to research questions about the solution's quality.

PS2.4 How do the countermeasures influence the outcome? Is the precision of the monitoring significantly affected? Are there other potential negative influences? These questions refer to the influence of the countermeasures on the monitoring solution's outcome and are therefore mentioned as the quality of the proposed mitigation. This influence must be mainly positive but may come with negative side effects which should to be identified and discussed.

PS2.5 What are the costs for the countermeasures in terms of operational costs, for example the bandwidth consumption, and computational costs, for example the complexity of the proposed algorithms? Beside the performance evaluation, the combined costs of the algorithms are of high interest. Both, the performance and its costs should be put into context and discussed.

PS3: Partial Participation in System-specific and Tree-based Monitoring Systems Research questions related to the partial participating problem in tree-based monitoring systems are discussed in the following. We aim to maximize the accuracy of the resulting global view while a subset of all nodes do not actively participate in the system. The global view must involve monitoring information about both, actively and passively participating nodes. First, we present questions about the status quo and the design of a potential solution.

PS3.1 Do state-of-the-art monitoring solutions suffer under a partial participation of peers? First, it should be investigated whether current solutions suffer under an incomplete participation and its implications should be discussed. This serves as a first proof of concept.

PS3.2 How can we organize active and passive nodes, so that the monitoring solution's global view considers both groups' system-specific monitoring information? This question refers to characteristics of a solution's idea and how passive peers can be involved in such.

Now, we refer to research questions concerning the quality and costs of a solution.

PS3.3 How does the solution influence the precision of the monitoring? This is one of the main questions as one may expect that the precision will be negatively affected. Therefore the analysis of the solution's influence on the precision is mandatory.

PS3.4 What are the solution's costs? Besides its performance and influence on the precision, the costs of a potential additional structure needs to be analyzed and put into context with the performance. An optimal system should adapt itself to the share of the non-participating peers, therefore, when all peers participate in the monitoring, its costs should be low or non-existent, respectively.

PS4: Peer-specific Monitoring in Peer-to-Peer Networks The fourth problem statement refers to the monitoring of peer-specific information. As the capacities might change frequently, for example the available bandwidth or CPU cycles, an infrastructure is needed to take this into account. Furthermore the search mechanism, providing a capacity-based search, should be fast and precise. In the following we point out research questions on this field. We start with those concerning the indexing structure of a solution.

PS4.1 How can capacities be indexed in an efficient way? Especially, what indexing schemes are suitable for high dynamic data? This central question distinguishes a potential solution for this problem statement and current solutions in the field of multidimensional indexing schemes in p2p networks.

A solution of this problem should be evaluated with a positive outcome under the following aspects.

PS4.2 How is the performance of this indexing scheme and its search? What about the precision and false positives of the search operation? The performance of the indexing structure should be oriented to adapt on capacity changes fast. The precision of the search should be high, optimally 100 %, which is the ratio of peers found in the queried range. At the same time, the false positive rate should be as low as zero, which means that the indexing scheme is able to cope with the capacity dynamism.

PS4.3 What effects do churn have on the solution? Churn, the peer dynamics, is an effect in p2p networks one need to take into account when designing protocols. Optimally, in such scenarios the solution should work reliably and should not lose much of its performance level.

PS4.4 How efficient is the indexing structure? What are the costs maintaining it? What are the costs of the search? This point refers to the structure's costs and costs for the search operation, which should be low. A good cost-benefit ratio is desirable.

PS4.5 How about its scalability? The scalability refers to the workload for the whole system and for each peer individually. Ideally it should scale with the number of peers and capacities.

PS5: Speeding up the Applicability of Peer-to-Peer Networks Lastly, we turn the attention to the poor accessibility of p2p applications and the accompanied number of users. Here we investigate to port p2p software to the web environment by using WebRTC. We expect to reach more potential participants by utilizing the easy-to-use web environment instead of a third-party software which is associated with the installation and configuration of it.

PS5.1 Is it worth porting (current) p2p software to the web platform using WebRTC? First, we need to evaluate whether it is worth to port a software to the web platform. The answer on the question "whether it's worth it" is difficult to answer. Before advising the community to port their software, which, depending on the project, might be a lengthy process, we need to build up a first prototype and evaluate it.

Our research questions referring the performance and costs of a solution are as follows.

PS5.2 Can a potential proof of concept and proof of work fulfill performance requirements? How is such port performing in comparison with a non-web environment? This question highlights the importance of the port's performance. Furthermore, it is important that the performance does not perceive a significant drop coming from the original.

PS5.3 What obstacles can be encountered? Because the web environment behaves differently, for example due to running in a sandbox, it may have a negative impact on the porting process, the implementation in general or, as mentioned earlier, its performance.

After having discussed the central research questions for our five Problem Statements PS1 – PS5 we refer to our contributions tackling these problems and their questions.

1.3 Contributions

In the following we outline our contributions by answering the arising research questions presented in the preceding Section 1.2. All solutions except those for PS5 were implemented in the p2p simulator PeerfactSim.KOM [44].

PS1: Increasing the Robustness in System-specific and Tree-based Monitoring Systems - SkyEye.KOM [45] and Mr.Tree [39] In the first problem statement, we aim on increasing the robustness of state-of-the-art tree-based monitoring solutions. These provide highly accurate communication structures but are prone to node dynamics which need to be eliminated by keeping the high accuracy and effectiveness. To answer Question PS1.1, in a nutshell, our first solution, which is an extended version of SkyEye.KOM, introduces new connections in the tree to be able to share information if only little information is available and the second solution, named Mr.Tree, introduces redundancy in terms of communication paths and data in the tree to be able to cope with high(er) churn. Questions PS1.2 – PS1.4 are being answered in the following by shortly introducing our contributions.

With SkyEye.KOM [45] we propose an extended version of the dissertation from Kalman Graffi [52]. SkyEye.KOM is a tree-based approach using a distributed hash table (DHT) to build up a monitoring tree running atop the p2p overlay as a separate layer. The local view of the peers is propagated in a bottom-up fashion by pushing aggregated data towards the root. The calculated global view is forwarded top-down piggybacked by acknowledgements (ACKs) as a response on the pushed local view. We propose to provide additional connections to siblings and data is pulled from them whenever little information is available, for example when changing the position in the tree. In the paper we evaluated this approach by conducting a parameter study and compared it with a gossip-based approach called Symmetric Push-Sum Protocol [14]. To answer Question PS1.2, we investigated the impact of churn. SkyEye.KOM is impacted by churn whereas the push-sum algorithm is barely affected. Still, SkyEye.KOM shows a higher precision and adaption rate although it dropped by 12 – 26 %. Regarding Question PS1.3 we summarize that higher costs are induced by SkyEye.KOM when encountering higher node dynamics but it is still under the level of push-sum. Thus, to this point, we answer Question PS1.4, which is the question whether the increased costs are worth the precision, with a weak yes.

After this, we evaluated SkyEye.KOM with higher node dynamics and encountered a weakness of peer starvation due to not getting any valid information and answered with a more robust solution, called multiple realities tree or Mr.Tree [39]. With Mr.Tree we enhanced the resilience of state-of-the-art tree-based aggregation schemes, which are tree-based monitoring solutions for example. The main idea is to build up multiple parallel trees and forming so called *hypernodes*. These hypernodes consist of all peers at the same position but in different trees and provide a transient storage for monitoring data. Whenever peers leave the network, the node which held this peer will still persist with a high probability as it consists of multiple peers. With different proposed modes we are able to dynamically build up new and prune unnecessary parallel trees. In the evaluation we answered Questions PS1.2 – PS1.4 as follows. When increasing the peer dynamics, we perceive an increasingly higher precision and a superior stable accuracy using our solution compared to the vanilla approach. We achieve a positive cost-benefit ratio with our 'adaptive' mode and in high peer dynamic scenarios we are able to reach high precision when using the 'fixed number of parallel trees' mode. Lastly we were able to improve the extended version of SkyEye.KOM when using the mechanisms of Mr.Tree in both, the accuracy and its costs.

PS2: Malicious Behavior in System-specific and Tree-based Monitoring Systems - Convex Hull Watchog [35] Our objective in this problem statement is to minimize the impact of maliciously behaving nodes on the accuracy of tree-based monitoring systems in the context of monitoring system-specific information. Regarding Questions PS2.1 and PS2.2, in our paper about Convex Hull Watchdog [35] we first investigated the impact of malicious behavior, like manipulation attacks. We see a devastating impact as even single peers can have a significant influence on the outcome of the monitoring. Especially, the counting to infinity problem, which is caused by loops in the tree structure, has a big impact. Therefore, we identified an attacker model including manipulation attacks, disruptive attacks against the tree's structure and free riding behavior. Each of the mentioned attacks is described and motivated. Concerning Question PS2.3 about the mitigation of such attacks, we propose a complete passive approach called DOMiNo to mitigate manipulation attacks by defining a convex hull using the Z-Median algorithm and only accept values inside this hull. Furthermore, we overcome a majority of structural attacks by defining expected origins in the tree structure. Passive approach means that we only sniff incoming packets and therefore do not exhibit additional costs, which answers Question PS2.5. Evaluation shows that an effective mitigation of manipulation and structural attacks is possible. We conclude that attackers are bound to a convex hull they can not calculate and therefore these attacks can be filtered out efficiently. Finally, structural attacks can be mitigated and the counting to infinity problem can be solved, which answers Question PS2.4.

PS3: Partial Participation in System-specific and Tree-based Monitoring Systems - Minicamp [37, 36] The objective of the partial participation problem is to keep a tree-based monitoring system intact and achieve an accurate system-wide monitoring while a group of nodes do not participate in the system. The first Question PS3.1, which refers to the need of a solution for the partial participation, can be answered in the affirmative. During the work on PS2 and our proposed solution Convex Hull Watchdog [35], we encountered a problem with peers refusing to participate in the monitoring service, which inevitably leads to eclipsed branches and therefore (partly) destroys the monitoring tree. This leads to a degradation of the monitoring solution's accuracy.

We propose Minicamp [37, 36] in a first outlook paper and a second detailed paper which includes algorithms and deeper evaluation. In order to tackle Question PS3.2, we propose a middleware which is located right between the p2p overlay and the monitoring overlay. The use of a middleware overlay makes it versatile and can thus be combined with various overlay classes running below and atop of it. Here, we map passive peers on active peers, whereby active peers are responsible for their passive peers as they need to discover them and scan their responsibility area for newly arrived peers. Furthermore, these active peers obtain the monitoring data from their passive peers and inject this data into the monitoring system located at the upper layer. Indeed, this structure is re-usable due to its nature of being a middleware. We propose to probe passive nodes to obtain monitoring data from passive peers, for example by initiating response time measurements. The evaluation shows that we are able to achieve a correctness of the node mapping of 90 % in a worst-case scenario with peer dynamics. The accuracy solely depends on the accuracy of the probing mechanism, which answers Question PS3.3. According to the costs indicated by Question PS3.4 we summarize that there are two cost sources, which are the maintenance of the middleware and the costs for the probing mechanisms. Prior to calculating the costs for the former source, we advise to merge the functionalities of the p2p overlay and our middleware to reduce the costs as proposed by Amft [3]. The latter highly depends on the probing mechanism itself and can therefore vary.

PS4: Peer-specific Monitoring in Peer-to-Peer Networks - CapSearch [34] and PacketSkip [32] We define the monitoring of peer-specific information in a decentralized environment as the gathering of peer-specific information in an appropriate structure and the dissemination in the form of a capacity-based search. Our goals are accurate gathering and dissemination processes while keeping the efficiency high. We contribute with two solutions on the problem statement of capacity-based indexing and search. Referring to Question PS4.1 we present two different ways on how to index the data. The first solution CapSearch [34] uses a kd-tree for a static multidimensional space partitioning and is a solution for Variant 1. Our second solution PacketSkip [32] projects the multidimensional data onto a one-dimensional plane and uses a Skip Graph [5] to efficiently search for data. It serves Variant 2, which is the dynamic and unbounded capacity structure. Both approaches are adapted to cope the dynamism of capacity data.

In the following we answer Questions PS4.2 – PS4.5 by referring to the solutions and their features.

CapSearch is our first approach to cope with the given problem statement. For CapSearch we require a fixed number of capacities and for each capacity a fixed number of classes, therefore it is a solution for the first variant. To operate this service we need a running DHT overlay. The tree nodes of the kd-tree are DHT objects and distributed throughout the DHT at well-defined places. The kd-tree is initially fully divided, so the leaves represent a single combination of capacity classes. Peers can access these tree nodes by performing a single DHT lookup, therefore the kd-tree structure can be quickly accessed and it is fail-safe due to a DHT replication service running in parallel. Inner nodes have a special meaning as they got a view of their sub-tree. Evaluation shows an optimal precision and false positives ratio in scenarios without churn. With churn, we encounter a high precision for the k-search (90 %) whereas the full-search degrades to 64 %. We verify a low and adjustable maintenance overhead and our approach scales logarithmically with the number of capacities and capacity classes.

In contrast to CapSearch, with PacketSkip we propose an approach for the second variant, which requires a dynamic number of capacities and an unbounded number of capacity classes. Similar to CapSearch, PacketSkip runs as a service atop a DHT overlay and the Skip Graph is realized by objects in the DHT. The multidimensional capacity data is mapped on the one-dimensional space of the Skip Graph, therefore we store labeled values in a one-dimensional space. Evaluation shows that PacketSkip scales with the number of peers but its workload distribution is worse than CapSearch as roughly every eighth peer stores an indexing structure object whereas in CapSearch almost every node (around 98 %) stores such an object. PacketSkip provides a superior capacity update mechanism, which is three to four times faster and needs five to six less messages per operation, and the search operation works almost perfect in terms of accuracy. Even under churn we encounter an accuracy of $\geqslant$ 98 % with a false positive rate below 2 %. For this superior performance PacketSkip pays the two- to four-fold maintenance overhead compared to CapSearch.

We conclude that PacketSkip and CapSearch serve different use-cases. CapSearch is the way to go, if the number of capacities is small and fixed, k-search is the search operation of preference, a very fair workload share is important and the costs should be as low as possible. In scenarios needing a high number or a dynamic number of capacities, PacketSkip is the better choice. Furthermore, PacketSkip should be chosen if full-search is often used.

PS5: Speeding up the Applicability of Peer-to-Peer Networks - WebP2P [33, 38] and Chunked-Swarm [97] Lastly, we present our contributions on the problem statement of the problematic accessibility of p2p systems. We answer Question PS5.1 by just attempting to go with the new standard WebRTC. Therefore, we made a proof of concept and a proof of work with our first two papers.

These papers present WebP2P, a browser-based overlay network with a simple social network operating atop. For this, we ported and adapted OpenChord[5] to use WebRTC with security modules and building blocks to support (video-) chatting applications. According to the main obstacles asked by Question PS5.3 we state that, when porting software to WebRTC, one needs to change to a single-threaded and event-based programing paradigm as browsers use an event queue. Furthermore, long calculations are not advisable as the GUI might freeze and the use of web-workers is therefore recommended. Back then, browser connections were hardly stable but nowadays the stability has been improved and these connections can be mentioned as stable. The signaling must be handled by the developer, which is a design decision in the WebRTC standardization. After all, the WebRTC-powered audio- and video-chatting is working fine, stable and is easy to implement.

Our third paper in this field is Chunked-Swarm. With Chunked-Swarm we propose an approach to smartly distribute chunked content between viewers interconnected in a mesh topology. Our target audience is a group of 200 viewers and is therefore recommended for smaller scenarios. The novelty is that we guarantee streaming delays which is new in the research field of p2p video streaming. However, this software had been implemented in its first version as a Java application, then several tests followed in the p2p simulator PeerfactSim.KOM and at the end this has been ported to the web using WebRTC as a successful startup. Therefore we can affirm Question PS5.2 as it works well in a real environment and it is not significantly disadvantaged compared to a standalone application.

[5]Project homepage of the OpenChord software: `http://open-chord.sourceforge.net/`

Lastly, we want to report that we got positive feedback from visitors at the Hanover Fair 2015 and 2016 on first prototypes of WebP2P and Chunked-Swarm, respectively, and are therefore optimistic that WebRTC is the way to go when running p2p systems in the near future.

1.4 Outline of this Thesis

In the following we outline this thesis' structure. We just introduced our work in Section 1 by stating the main motivation on p2p networks and the monitoring of p2p networks along with problem statements in both fields. An overview with research questions and a summary of our contributions answering these questions is given. The rest of this thesis is structured as follows. We give a broader motivation and detail our solutions for the Problem Statements PS1 – PS5. These are handled in the Sections 2 to 6. In these sections each of the ten papers is summarized, related works, contributions, personal contributions and the impact on this thesis is given and discussed. In particular, these sections are structured as follows.

In Section 2 we give an introduction on decentralized monitoring solutions in general and on the enhancement of tree-based monitoring systems by increasing their robustness in particular. We present our first solution SkyEye.KOM [45] in Section 2.1 and we detail and motivate the need of our second solution Mr.Tree [39] in Section 2.2. In an appendix in Section 2.2.6 we compare the solutions Mr.Tree and SkyEye.KOM in more detail.

The motivation of malicious behavior and its influence on p2p monitoring solutions is presented in Section 3. Following this, we present our paper Convex Hull Watchdog [35] with a detailed motivation for an attacker model and our mitigation mechanism DOMiNo in Section 3.1.

The problem statement about the partial participation of peers is motivated in Section 4. Our solution Minicamp is presented in two papers. The first is Minicamp [37] and states the problem statement and presents an overview of this problem, which is done in Section 4.1. An extended version is given in our paper Minicamp [36] and is detailed in Section 4.2.

In the following Section 5 we deal with the problem statement of indexing and search of peer capacities and motivate this field of research. Our first solution CapSearch [34] is presented in Section 5.1. PacketSkip [32], our second solution in this field, is stated in Section 5.2.

Section 6 refers to the problem statement about the bad applicability of p2p systems and the role of WebRTC to enhance it. In this section we motivate the need of getting rid of third-party application installations if a user wants to access a p2p system and motivate the open standard WebRTC as a solution for this problem. First, we present our demonstrator WebP2P [33] in Section 6.1. In Section 6.2 we present another paper concerning WebP2P [38] and give more details about the platform and conduct an evaluation of it. With Chunked-Swarm [97] in Section 6.3, we present a third paper on this topic, which primarily guarantees streaming delays but also has had the potential to be operated in the web by using WebRTC.

Finally, we conclude our work, give thoughts about future work and give closing remarks in Section 7.

Chapter 2

Increasing Robustness of structured Monitoring Approaches Collecting System-specific Data

In the first chapter after the introduction of this dissertation we refer to the crucial tasks a monitoring approach has and we stress its underestimated function in a communication network. Especially in decentralized networks, applications running atop a communication network, for example LifeSocial [46], require a stable and reliable underlaying network infrastructure, for example Pastry [104]. These applications provide users a certain level of *quality of service* (QoS) to increase the *user experience* (UX). Hence, participants of such applications anticipate a fluid user interface (UI) with fast response times. Hossfeld et al. [58] quantify the impact of initial waiting time and waiting time in general on the quality of experience (QoE) users perceive. The authors conclude that users are sensitive to any interruptions during runtime and they propose to design applications accordingly and minimize interruptions. To fulfill this requirement a precise and up-to-date monitoring system is mandatory for the network infrastructure to adapt itself according to the current network conditions. Monitoring is hence defined as a service in a communication network which gathers local information from its participants and provides an evaluated view to them. There exist different monitoring classes which either continuously (proactively) or on demand (reactively) gather the information. Furthermore, the evaluated view is either situated, concentrating on only a sub set of the network, for example the two-hop neighborhood, or global, including all nodes in the network (system-wide). The evaluated view can be either pushed automatically to the participating nodes or the nodes need to obtain the view by pulling it. Lastly, the information to be monitored needs to be defined. In this chapter we focus on system-specific monitoring data, that is data related to the system itself. Examples for metrics to measure a system's state are the hop count needed to finish a lookup, the number of sent bytes on the overlay layer and the delay for a store operation of an object.

For each metric that can be used to evaluate the status of the system a set of aggregatable attributes is defined. These aggregatable attributes or functions, respectively, are for example the count, minimum, maximum, mean and variance. Thus, for each monitoring metric the above mentioned aggregatable functions are applied. Using aggregated data in a monitoring system has the main advantage that it is independent on the actual number of data sets gathered, therefore the system scales perfectly and gathered data has always the same size, which is the number of metrics multiplied with the number of applied aggregatable attributes.

Summarizing, a system-specific monitoring solution collects information about the network structure and network conditions, so a controlling unit is able to react on recent changes. Therefore, the controlling unit and often many further mechanisms in a p2p system are highly dependent on accurate data from the monitoring solution.

State-of-the-art monitoring solutions can be categorized into *structured* and *unstructured* monitoring systems. A detailed survey is given by Makhloufi et al. [79]. The class of unstructured monitoring solutions concentrate on gossip-based communication between participants and therefore no particular responsibilities are intended. Here, participants share their monitoring data with their neighbors or random contacts. By doing this, the local view of each participant approximates the global view and the local view is reseted after a pre-configured number of exchanges. This period of time is called an *epoch*. At the beginning of each epoch, each participant surveys its local monitoring data and uses it for the further exchange process. Whereas, at the end of each epoch the local view represents the approximated global view and is therefore most precise. Examples for unstructured monitoring approaches are the Push-Sum Protocol [69], its successor Symmetric Push-Sum Protocol [14] and T-Man [65]. On the other hand, in structured monitoring approaches participants follow a well-defined responsibility pattern to collect and disseminate monitoring data. Most approaches use a tree-based structure to collect data up and down the tree, respectively, in $\mathcal{O}(N)$, where N is the number of participants in the monitoring solution. Prominent solutions are SingleTree [10], SOMO [137], SDIMS [131], SkyEye.KOM [50] and GAP [27] with its extensions A-GAP [96], (P)M-GAP [127] and TCA-GAP [126].

In general, the two classes mentioned before are either highly robust or very accurate. On the one hand, structured solutions are very precise but fragile due to disruptions in the tree structure, which is evaluated by Makhloufi et al. [79]. Unstructured gossip-based solutions on the other hand are very robust as they do not follow any structure but are not as precise and up-to-date as structured approaches. So, the decision between these solutions depends on the special use-case: is a high fluctuation of the participants to be reckoned with? Do participants rely on exact values? In this chapter of the dissertation we propose structured monitoring solutions if one has to answer both questions in the affirmative. Actually, most real use-cases fall into this category, as churn is a natural phenomenon in p2p systems and a monitoring system should aim for a high precision.

In the following we present two papers: an extended version of SkyEye.KOM [45] and Mr.Tree [39]. The former paper is presented in Section 2.1, where we propose a structured monitoring approach called SkyEye.KOM with extensions to enhance its robustness. This is mainly achieved by introducing gossip connections between siblings in the tree. The latter paper is presented in Section 2.2, where we stress the tree-based monitoring approach even further, reveal an important problem and propose a solution which is based on the duplication of data and paths in the communication structure.

2.1 SkyEye: A tree-based peer-to-peer monitoring approach

In this section we state our contribution, discuss related work and summarize our extended paper about SkyEye.KOM [45], which is appended as a verbatim copy at the end of this section.

Kalman Graffi and Andreas Disterhöft. "SkyEye: A tree-based peer-to-peer monitoring approach". In: *Elsevier Pervasive and Mobile Computing, Volume 40*. 2017.
Impact Factor (2016 - two preceding years): 2.349.

In the following, we first state our contribution in the field of decentralized monitoring solutions in Section 2.1.1 and then refer to the importance and impact on this thesis in Section 2.1.2. Afterwards we summarize our paper in Section 2.1.3 and discuss related literature in Section 2.1.4. We close this part with personal contributions in Section 2.1.5.

2.1.1 Contribution

The contribution of this paper is presented in the following and is divided into three parts.

First, we present the proposed decentralized monitoring solution SkyEye.KOM in detail. This point covers the creation and maintaining of the tree-based monitoring topology and its proactively push-based collection of system-specific monitoring data and the dissemination of the global view. We propose mechanisms to tackle robustness issues of the tree that introduce additional interconnections in the tree. These are used to obtain data in times of spares information. SkyEye.KOM creates its own monitoring overlay atop a running structured p2p overlay and is therefore versatile. The underlying structured p2p overlay has to implement the KBR interface [25]. Possible p2p overlays are therefore Chord [115] or Pastry [104].

Second, we performed an extensive parameter study of our SkyEye.KOM and Symmetric Push-Sum implementation and present the most interesting findings. Focusing on SkyEye.KOM, we varied the update interval, branching factor and investigated the influence of different underlying overlays. By decreasing the update interval, the precision and freshness of the global view decreases but the costs are increased due to more frequent information exchange. The same applies for increasing branching factor with the addition that the cost distribution throughout the peers becomes more unequally. This is mainly due to the fact that an increasing branching factor influences the relation between the number of leaf and inner nodes of the tree. Therefore more leaf nodes are present who do not utilize their download channel to collect monitoring data up the tree. The influence of different underlying p2p overlays on the monitoring performance is not critical. Due to less overlay costs and faster routing in the practice, Pastry performs slightly better than Chord.

Third and last, we compared the performances of both approaches, SkyEye.KOM and Symmetric Push-Sum, under churn. We conclude that our proposed approach is negatively affected by

churn. On the other side, we verified that the Push-Sum implementation is only barely affected by churn. Nevertheless, our proposed approach outperforms the Push-Sum implementation by far in terms of accuracy, adaptivity and costs.

2.1.2 Importance and Impact on Thesis

With the paper about SkyEye.KOM [45] we accomplished a fundamental building block for this thesis. The vanilla implementation in PeerfactSim.KOM is used and extended by further mechanisms throughout this thesis. It is used as a representative for the class of proactive push- and tree-based monitoring systems which gathers a common global view. The terminology is taken from Makhloufi et al. [79].

The proposed steps to enhance the robustness of the fragile tree-based monitoring class is an essential first step for our paper Mr.Tree [39], which is presented in Section 2.2. Mr.Tree can be built atop SkyEye.KOM to further enhance the robustness for scenarios facing very strong churn.

Besides the work on fixing the lag of robustness in tree-based monitoring systems, we investigated malicious nodes in such environments. Our solution Convex Hull Watchdog [35], presented in Section 3.1, uses SkyEye.KOM to disclose attack vectors on such systems and proposes a mitigation of such.

Finally, we analyzed scenarios where only a part of all nodes participate in the monitoring overlay in Section 4. We propose Minicamp [37], which builds up a middleware overlay and is positioned between the p2p overlay and the monitoring overlay. For the latter we used our fundamental building block, the implementation of SkyEye.KOM.

2.1.3 Paper Summary

In this journal paper we present an extended version of SkyEye.KOM, which is a decentralized structured monitoring system. It is used to gather system-specific information of the underlying structured p2p system. This information, which is referred as a monitoring metric, might be the average hop count needed to lookup a certain ID in the overlay or the average response time for the whole lookup process. The underlying structured p2p system must implement the key-based routing (KBR) interface [25]. SkyEye.KOM creates a *new layer* atop this KBR-compliant p2p overlay instead of using an integrated p2p overlay and monitoring approach.

The main idea of the system-specific monitoring data gathering is to build up a tree with nodes being the peers who participate in the underlying p2p system. Monitoring data is gathered bottom-up by aggregating and passing it to the parent node, thus, only one data set is needed for each monitoring metric. The data set aggregated at the top of the tree, at the root node, is called the global view as it contains all aggregated data of all nodes. This global view is propagated top-down by using piggybacked acknowledgements (ACK). In the following, we go into more protocol details. First, we refer to Figure 2.1 and describe how nodes are positioned in the monitoring tree. For independent node addressing, the underlying p2p overlay space is

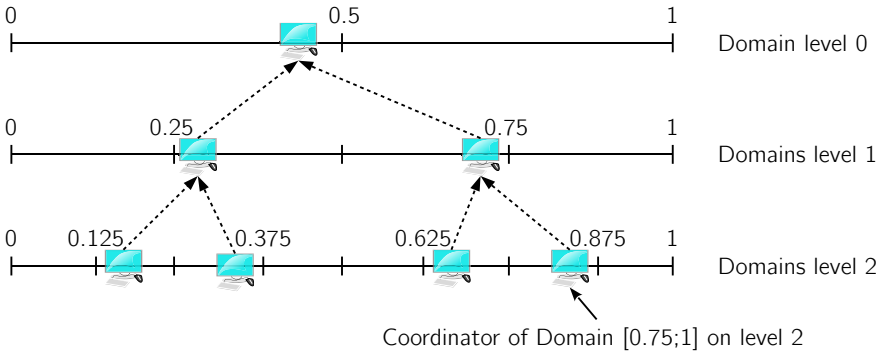


Figure 2.1: Example of a binary SkyEye.KOM tree till level 2. Coordinators, which are peers responsible for the appropriate domain, are shown together with their next hop on the bottom-up communication path.

mapped onto a unified ID space $[0;1]$. The k-ary tree is built up by subdividing the unified ID space recursively into k subregions, which are called *domains*. Therefore, the whole unified ID space is the domain on level 0. In general, domain i is subdivided into k new domains which form domains on level $i+1$. The *coordinator* of a domain is defined as the node which is responsible for the ID in the middle of the domain or ID range, respectively. Taken all together, the tree structure consists of domains which are nested level-wise. Now, if a node wants to find its position in the tree, it performs the following steps. First, it maps its p2p overlay node ID onto the unified space. Then, it checks whether it is the coordinator of the domain on level 0. If it is responsible for this ID, it becomes the root node of the tree. Otherwise, it goes on to level 1 and proceed to the domain where the mapped node ID lies in. Then, it tests whether it is the coordinator for this domain, if not, it proceeds to level 2 and performs the same steps as mentioned before. Eventually the node finds its position in the monitoring tree and is ready to send monitoring data. So, secondly, we describe how the monitoring data flows along the tree. Each node is coordinator of a domain and due to the deterministic structure of the tree, each node can calculate its parent in the tree structure. This can be performed by calculating the coordinator of the domain it was divided from. Thus, a node is able to calculate its parent ID, perform a lookup for this ID and send the monitoring data to the node responsible for this ID. Thirdly, we give an outline of the tasks each node starts when a new propagation period begins. SkyEye.KOM follows a proactive approach, thus, data is sent periodically. The time between two periods is therefore called *update interval* (UI). At the beginning of each update period, each node performs the following steps:

1. calculate its position in the monitoring tree

2. aggregate all received child estimations of its local data base and its local estimations and

3. send the data to the calculated parent up the tree.

The root node, which is the node responsible for the single domain on level 0, aggregates its gathered child estimations and its local view to the global view. This global view is propagated top-down by piggybacking the global view with an ACK which is sent to children nodes when

receiving valid monitoring information. By doing this, aggregated data flows periodically bottom-up and the global view is propagated top-down as a response on successful received aggregated monitoring data.

Lastly, we provide information on how the robustness of SkyEye.KOM is improved. We start with a small recap on why SkyEye.KOM, and decentralized, structured and tree-based monitoring approaches in general, are prone to the joining and leaving of nodes (churn). When nodes join and leave the network, responsibilities in the underlying structured p2p overlay change. As a result, nodes change their position in the monitoring tree, because the calculation of a node's position in the tree depends on the responsibilities in the p2p overlay. An ever-changing tree is problematic as a node needs to reset its data base whenever it changes its position. This results in a loss of data of the whole underneath sub-tree; a devastating effect. In this paper we propose additional connections in the monitoring tree: between siblings and between old and new coordinators. We extended SkyEye.KOM with mechanisms to mitigate the effect churn has on the tree structure. These mitigation mechanisms are the following:

- We propose the policy 'to not pollute the monitoring tree with outdated or insufficient data'. We assume that the monitoring tree is balanced and the sub-trees of each level do not divergate too much from each other. Therefore, nodes, which just changed their position in the tree and have therefore insufficient data to forward it up the tree, ask their siblings instead to obtain first estimations. For our mechanism, we added states to the nodes: stable and unstable. Whenever a node just changed its position in the tree, it becomes unstable. After two update intervals (2*UI) it becomes stable again. In the stable state the vanilla SkyEye.KOM protocol is executed. In unstable state, the node does not forward incomplete monitoring data as the node still waits for children to push their data. Instead, to get a first estimation, we propose to ask siblings on the same level (brothers and cousins) for their monitoring data. Siblings on the other hand only answer with their current monitoring view if they are in a stable state. On receiving such responses, we take the average and use this to provide new information to the node's parent. Ideally, after one UI, fresh information from the children nodes arrive and from then on the local node can use this information. This mechanism is only enabled for levels high in the tree as they have an important role for the precision of the data. Remember, if these nodes leave, the data of this sub-tree is lost. To limit the bandwidth consumption, not all siblings are asked, but just k at maximum, with k being configurable.

- We propose a mechanism to delegate a node's old data base to the new coordinator of the domain it was responsible for, so that the new coordinator can start with a, probably partly outdated, data base. For this, whenever a node changes its position, it sends its current monitoring data base to the new coordinator which can continue with this data base.

In order to operate on information as fresh as possible, nodes delete monitoring data, which are older than $4 * UI$, typically after one to two minutes. Therefore, the timing for the mitigation mechanisms is very important as the parent node might starve, and therefore delete a sub-tree, if we were not able to provide fresh monitoring information.

After the elaboration of the SkyEye.KOM protocol, we classify our approach according to the model from Makhloufi et al. [79].

- *Network view*: We applied the global network view as all peers participate in the monitoring approach and all peers receive the global view.
- *Network structure*: SkyEye.KOM follows a structured tree-based network topology.
- *Propagation policy*: Monitoring information flows proactively through the tree.
- *Neighborhood information*: This approach is in the class of *informed* as each node exchanges its monitoring information with a dedicated neighbor or a dedicated sibling. Thus, the data flow is deterministic and given by the tree structure.

In the evaluation we compare our implementations of SkyEye.KOM and the Symmetric Push-Sum Protocol, which is proposed by Blasa et al. [14], in the PeerfactSim.KOM p2p simulator framework [44]. The metrics, we are interested in, are the precision and freshness of the global view and the costs of these approaches. The analysis consists of three parts. First, we perform a parameter study of both approaches. Here, we evaluate different update interval lengths, branching factors for SkyEye.KOM or rounds per epoch for the Push-Sum Protocol, respectively. Second, we set the best parameters for SkyEye.KOM and evaluate the influence of a different overlay. The impact is limited and due to better lookup performance. With the evaluated best parameters for both approaches, we compare both approaches atop the structured overlay Chord [115]. SkyEye.KOM proves to be the better choice due to a superior precision, adaption rate and less monitoring costs for SkyEye.KOM. Lastly, we investigate the impact of KAD churn, a churn model created by Steiner et al. [113] from the churn behavior of peers in the file-sharing application eDonkey. We could not recognize any significant impact on the Push-Sum implementation, which is the main argument to choose this approach. For SkyEye.KOM we notice a slight decrease in precision but it is still higher than the precision of the gossip-based approach. Furthermore, we observe worse adaptation rates than without churn. For the monitoring layer bandwidth we observe that SkyEye.KOM faces an increase of up to 100 %, whereas Push-Sum perceives an increase of roughly 33 %. Still, the Push-Sum Protocol induces roughly three times more monitoring costs than SkyEye.KOM. For the bandwidth on the network layer, we observe an increase of 12 – 26 % under churn for SkyEye.KOM. For Push-Sum on the other hand, we only observe an average increase of 4 %. However, our implementation of SkyEye.KOM perceives 36 – 40 % more traffic on the network layer than Push-Sum, because each peer needs to perform one lookup each time it sends data up the tree, which is set to every 30 seconds. A peer caching mechanism would significantly lower these costs, especially in scenarios without churn. The Push-Sum Protocol implements such a peer cache mechanism and furthermore each peer starts with a set of peers from their DHT neighborhood, therefore DHT lookups are a rarity here. One need to note that doubling the update interval of SkyEye.KOM to 60 seconds causes the network layer costs to be halved, even without the use of caches, while still outperforming Push-Sum in terms of precision and adaption rate. Thus, SkyEye.KOM is the superior choice, even when facing realistic churn behavior.

2.1.4 Related Work on Decentralized Monitoring Solutions

In the last decade a lot of work in the field of decentralized monitoring or aggregated information had been done. In this section we elaborate work related to SkyEye.KOM in classes

according to the four fields used in the survey from Makhloufi et al. [79]. These are the network structure, propagation policy, network view and neighborhood information. Solutions mentioned in this dissertation are categorized and shown in Figure 2.2. In the following, we denote the class membership as a four-tuple. We start with work close to SkyEye.KOM, thus, with work located in the same class.

As mentioned before, the class of SkyEye.KOM is ⟨tree, proactive, global, informed⟩. SOMO [137] creates a metadata overlay where nodes, like in SkyEye.KOM, build up a tree and the information flow is bottom-up for aggregating data and top-down for the global view. In contrast to SkyEye.KOM, SOMO is pull-based and therefore parents ask their children for data. This induces a certain overhead and a tree prune operation is needed which is triggered by an additional periodical check. SDIMS [131] uses the Astrolabe system proposed by Van Renesse et al. [119] to construct the hierarchy of nodes used for their tree structure. Furthermore, SDIMS uses multiple trees with one tree per attribute. The authors state a problem of the load distribution one-tree solutions have: Leaf nodes have a lower load than intermediate nodes as they do not have children who push data to them. The authors argue that a multiple-tree structure leads to a more evenly load distribution as nodes are in some trees leaves and in others intermediate nodes. However, the complex structure leads to complex queries when requesting multiple attributes. A-GAP [96] and TCA-GAP [126], two solutions that base on the GAP [27] protocol, are worth mentioning as they are located in the class of SkyEye.KOM, too. However, first, we refer to GAP which is located in the class ⟨tree, proactive, global, blind⟩. GAP is a very fundamental approach and builds up a breath-first search (BFS) spanning tree which contains all nodes participating in the underlying network. The aggregate data is periodically sent bottom-up, thus, data is handed over to the node which contacted a node first in the BFS operation. Nodes in the tree keep track of their neighborhood and all other nodes which contacted them and collect information about them. When facing churn, it tends to be quite challenging to repair such a tree structure as the positioning of nodes is more or less random. Whereas, in SkyEye.KOM, positions in the tree are bound to the responsibility of the underlying DHT, thus, the positioning is clear and definite. Referring to A-GAP, it additionally introduces an adaptation mechanism which aims to save bandwidth consumption. Here, nodes do not send updates on each period, instead, they forward data only if a certain local filter is exceeded. In TCA-GAP a threshold crossing alert system is used for the monitoring. On reaching a certain threshold, e.g. for a metric such as hop count, nodes become active and start to exchange aggregated monitoring data, otherwise they enter a passive mode and cease to forward data up the tree. The root of a (sub-)tree decides whether the nodes underneath need to become active or not.

The next class we refer to is ⟨tree, proactive, situated, blind⟩, whose solutions build up a situated aggregation, which means that nodes get a situated view of only areas of the network instead of getting a system-wide (global) view. This situated aggregation can be distributed instead of or in addition to a global one. The authors of M-GAP [127], an extension of GAP, proposed the latter. Instead of sending partial aggregations of a whole sub tree up the tree towards the root, in M-GAP, nodes get all partial aggregations from their children and hand them over to their parent. Thus, instead of a single aggregation, a vector is sent. The need of such a mechanism highly depends on the use-case and increases the operational costs due to larger messages to be exchanged. Therefore, we focus on a global view only in SkyEye.KOM.

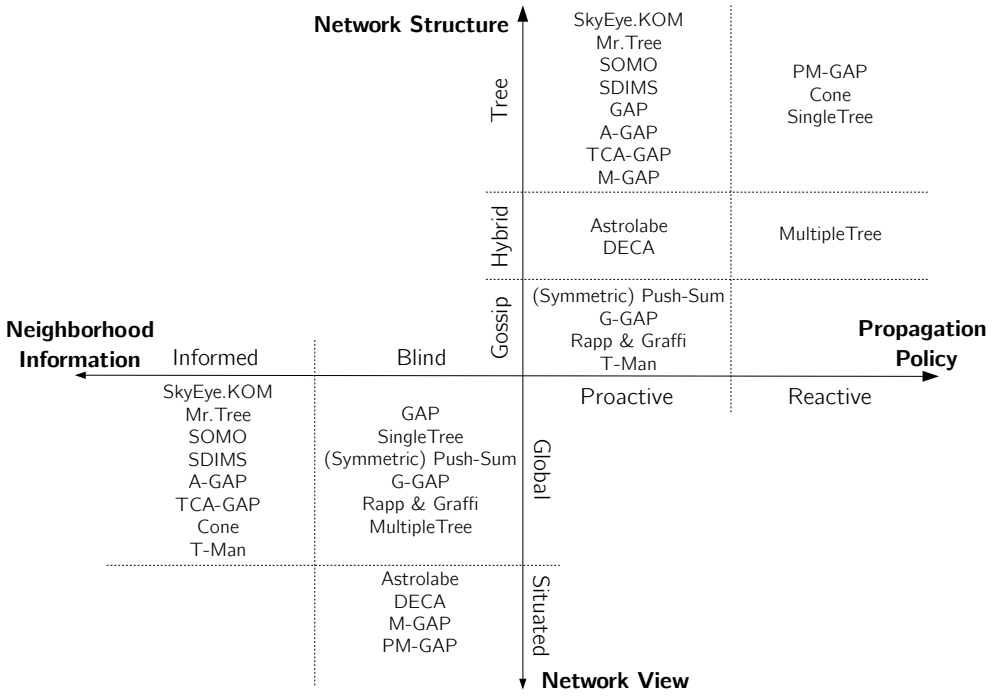


Figure 2.2: Related work discussed in this dissertation about system-specific monitoring solutions. These solutions are categorized into the network structure, propagation policy, network view and neighborhood information. This figure is based on Figure 1 from Makhloufi et al. [80].

Now we hand over from proactive to reactive approaches, thus, having a closer look on solutions of the class ⟨tree, reactive, *, *⟩. In this context an asterisk means *any*. PM-GAP [127], which extends M-GAP, uses a pull-based mechanism for the information gathering. By doing this, nodes do not get overloaded due to frequent changes of their data. This might be implemented in a periodic operation, and therefore remaining proactive, or can be interpreted to become a reactive solution. Cone [12] uses a multi-tree structure, one tree per aggregate function and attribute and is used for resource discovery. These follow a heap ordering and are therefore unbalanced and mentioned as *tries*. Nodes can search for certain values in such tries, for example values above 100 of the resource power. Aggregation of information in a trie is performed in a reactive manner. A traditional reactive approach is SingleTree [10]. Nodes in SingleTree build up a temporal BFS spanning tree if a node initiates it via a query. The initiator is the root node and once the whole spanning tree has been built up, leaf nodes start to send their monitoring information up the tree towards their parents. Intermediate nodes anticipate that data from all of their active children have been pushed before forwarding the aggregated data to their parents. Eventually, aggregated data reaches the root who can build its global view and the temporal spanning tree dissolves. The choice between a reactive and proactive approach is actually a design decision. If continuous monitoring is not required for an application, a reactive approach might be a better choice. In SkyEye.KOM, however, we have chosen a push-based proactive approach, so that all nodes are continuously kept up-to-date and the overhead is minimized by pushing information instead of pulling it.

After having a closer look on related work in the tree-based monitoring approaches, in the following we move to gossip-based approaches, more specifically, the class of ⟨gossip, proactive, global, *⟩. The Push-Sum Protocol from Kempe et al. [69] is one of the fundamental work in the field of gossip-based monitoring approaches. They propose a mechanism that can run atop arbitrary networks where neighboring nodes periodically exchange their views, i.e. by gossiping about monitoring information. The communication is epoch-based consisting of several 'gossip' rounds per epoch and eventually, at the end of an epoch, each node holds the approximated global view. Its successor, proposed by Blasa et al., is the Symmetric Push-Sum Protocol [14], which enhances the performance by speeding up the convergence speed. G-GAP [128], which is based on the Push-Sum Protocol, enhances its robustness against mass node loss in the neighborhood. It introduces a control mechanism to trade-off between estimation accuracy and protocol overhead. Another important contribution is T-Man [65] which builds whole overlays with gossiping approaches and provides therefore a very generic set of operations. The authors of T-Man built for example Chord-based distributed hash tables with their approach. An epoch-less gossip-based approach is proposed by Rapp and Graffi [99]. The authors propose to continuously add a fraction of new data per gossip round. By doing so, old data is forgotten and the global view converges to the current view quickly with a certain calculable error. However, gossip-based approaches can be implemented in versatile environments and they are naturally very robust against churn. But their precision is very limited, due to the method to approximate the global view.

Lastly, we want to call attention to a *hybrid* solution from the class ⟨hybrid, reactive, global, blind⟩. MultipleTree [10], which is an extension of SingleTree, is an interesting approach to enhance the robustness of tree-based approaches by introducing redundancy. Instead of building just one temporal BFS tree, the authors propose to build k independent temporal BFS trees. Independent means that nodes should not be placed in the same position in two trees. In this solution the query traverses the tree bottom-up and each node waits to get answers of all active children in all trees. Therefore, the probability to lose a whole sub-tree, due to a churn event, shrinks as of the diversity of the trees. In this journal paper we introduce a mechanism to mitigate the impact of churn by smartly forwarding and requesting additional data. In Mr.Tree [39] we introduce a redundancy mechanism for solutions such as SkyEye.KOM in order to overcome even strong churn. In Section 2.2 we explain Mr.Tree in more detail.

2.1.5 Personal Contribution

The main source of content for this paper is provided by the first author Kalman Graffi with his dissertation entitled 'Monitoring and Management of Peer-to-Peer Systems' [52]. Andreas Disterhöft, the author of this thesis, extracted the system-specific monitoring approach SkyEye.KOM and condensed its content for this paper. He re-wrote the motivation, evaluation and conclusion. He added the mechanism of additional connections between siblings and two generations of coordinators to enhance the robustness of the monitoring tree and changed Figure 3 accordingly. He started with the implementation from Philipp Giesen and re-simulated all scenarios for the evaluation. Kalman Graffi gave valuable reviews and discussed general issues. Phillip Giesen, who did not actively participate in this paper, implemented and evaluated the vanilla version of SkyEye.KOM in the p2p simulator PeerfactSim.KOM during his Master's thesis, which was used in this paper.

2.2 Mr.Tree: Multiple Realities in Tree-based Monitoring Overlays for Peer-to-Peer Networks

This chapter contains the contribution, a summary and related work of our paper Mr.Tree [39], which is appended as a verbatim copy.

> Andreas Disterhöft, Phillip Sandkühler, Andre Ippisch and Kalman Graffi. "Mr.Tree: Multiple Realities in Tree-based Monitoring Overlays for Peer-to-Peer Networks". In: *Proceedings of the IEEE International Conference on Computing, Networking and Communications (ICNC)*. 2018. Acceptance Rate: 26 %.

We start with our main contribution, which is stated in Section 2.2.1, and elaborate its importance and impact on this thesis in Section 2.2.2. A more detailed paper summary is given in Section 2.2.3 and after that we discuss related work on the resilience problem of tree-based monitoring solutions in Section 2.2.4. Lastly, in Section 2.2.5 we disclose the personal contributions.

2.2.1 Contribution

In our paper we motivate the need for resilient tree-based monitoring systems, propose a solution called Mr.Tree, which can be adapted for existing approaches, and evaluate its performance by implementing a version atop SkyEye.KOM.

Mr.Tree is a generic aggregation scheme protocol to enhance the resilience of state-of-the-art tree-based monitoring systems. More precisely, it is suitable for solutions of the class ⟨tree, proactive, global, *⟩. This class contains, but is not all inclusive, the following solutions: SkyEye.KOM [45], SDIMS [131], SOMO [137], GAP [27] and its extensions A-GAP [96] and TCA-GAP [126]. Additionally, the monitoring solution must operate atop a common structured p2p overlay which is KBR-compliant [25] such as Chord [115] or Pastry [104]. In a nutshell, Mr.Tree provides mechanisms to build up multiple parallel trees and nodes at the same logical positions in different trees maintaining a transient storage for monitoring data. This shared storage is used to select the most likely data set for further processing. By doing this, we are able to increase the accuracy of the global view. We implemented a prototype atop SkyEye.KOM in the p2p simulator framework PeerfactSim.KOM. Lastly, we performed an extensive evaluation by investigating the behavioral, precision and cost differences between the vanilla and Mr.Tree-powered implementations while increasing node fluctuations in the network.

2.2.2 Importance and Impact on Thesis

Our paper Mr.Tree is a very important work and has a high impact on this thesis. We made a strong contribution to a known problem with our proposed approach on mitigating the

natural fragility of structured tree-based monitoring approaches, namely, solutions located in the class ⟨tree, proactive, global, *⟩. With the proposed mechanisms of the extended version of SkyEye.KOM, we presented earlier in Section 2.1, the precision and cost were fine in scenarios with low churn but these mechanisms are not suitable for scenarios facing higher churn. In these scenarios we encounter a high loss in precision, mainly due to starved parent nodes positioned high in the tree. This means that parents do not get any further data because their children are waiting for data from their children and so on. This results into an erase of data after a certain time. Thus, the proposed additional connections between siblings and the information recover mechanisms of replaced coordinators are not sufficient when facing high node fluctuations. Therefore we have presented Mr.Tree, a logical successor to solve this problem by introducing a configurable amount of redundancy while remaining cost-efficient. The objective to enhance the robustness of structured monitoring approaches in scenarios of low, through moderate, to high churn is therefore successfully completed, which concludes the first part of this thesis.

2.2.3 Paper Summary

In our paper Mr.Tree, which is the abbreviation for Multiple Realities TREE, we tackle the robustness problem tree-based monitoring solutions have. According to Makhloufi et al. [79] and Jesus et al. [66] such approaches are highly accurate on aggregating data, such as monitoring information, but are fragile when nodes departure from or join the network. Thus, they are prone to churn and have severe issues with node fluctuations during runtime. Makhloufi et al. [79] evaluated this behavior and verified it for GAP [27]. In Section 2.1 we already presented mechanisms to mitigate the impact node fluctuations have on the tree structure. However, the mitigation mechanisms are only able to achieve high accuracy, if the churn behavior is fairly low, thus, nodes occasionally depart and join the network. When the node fluctuation rises, the 'do not pollute the network with wrong information'-policy leads to a starvation of nodes which ultimately leads to data loss from sub-trees. We support this statement in Section 2.2.6, where we evaluate and compare these approaches. However, in this paper we propose a solution that keeps the accuracy up, even in times of high churn.

We propose Mr.Tree, an extension for a group of state-of-the-art monitoring approaches, where additional redundancy and data replication is achieved by building up multiple trees or by introducing logical hypernodes, respectively, which replace single physical nodes in the former approach. In the following we define *nodes* as logical hypernodes and *peers* as physical nodes. Peers in the same position, which are all peers of a node, share their data and therefore provide a transient data base. Hence, we are able to decrease the probability of data loss, which is caused by failing nodes. Furthermore, we can adjust the fail probability as there is a correlation between the size of logical nodes, which conforms to the degree of replication, and the fail probability of data. In the following we explain the protocol of Mr.Tree in more detail.

Figure 2.3 gives an overview of the hypernode structure and important mechanisms. The original protocol, which is extended by Mr.Tree, is executed in the first dimension or reality, respectively. These peers, which are called *coordinators*, recruit other peers to create further dimensions, thus, they tell other peers to create another instance of the original tree. The term coordinator is inspired by the terminology of SkyEye.KOM. Peers in dimensions $i > 1$ are therefore called *parallel peers*. For efficiency reasons, leaf nodes of the tree do not recruit

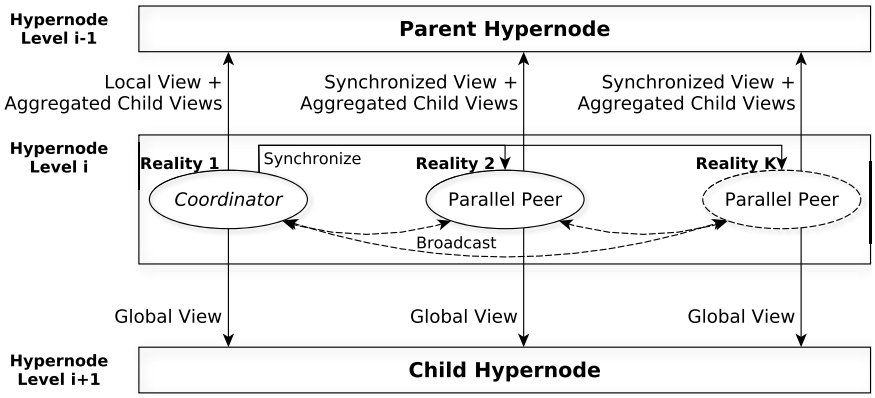


Figure 2.3: Overview of the hypernode structure. The hypernode on level i communicates with the hypernode on level $i-1$ and $i+1$ to push aggregated views or the current global view, respectively. Inside the hypernode, the coordinator is the peer responsible for this node in reality one and the other peers are mentioned as parallel peers. Views are exchanged by broadcasting them inside the hypernode and the coordinator synchronizes the parallel peers of the realities $i > 1$. This figure is taken from our paper Mr.Tree [39].

parallel peers as, in a case of failure, only their single monitoring data set is dropped. Parallel peers of the i^{th} dimension build up a tree, which is independent of the other dimensions, and follows the protocol of the original tree but with different peers on same positions with a high probability. Peers on same positions in different dimensions communicate with each other and create a logical node. Inside logical nodes, two mechanisms are proposed to handle the synchronization and communication between these dimensions. Firstly, the timing: The peer in the first dimension broadcasts its monitoring data at the beginning of each interval. This is also used as an implicit recruiting of new peers. Secondly, the data: Each time any of the peers of a node receive data relevant for the monitoring, it broadcasts this information to all other peers in other dimensions. By doing this, all peers in a node receive all data, thus, all peers are always kept up-to-date. It is important to note that peers do not inject their local measured monitoring data in dimensions other than the first. Hence, parallel peers replicate local monitoring information of the node in the first dimension, aggregate it with data received from their children nodes and help to distribute the data, if required. With Mr.Tree we propose three protocol modes, each following a principle:

1. *Fixed* - 'simple but robust': The number of dimensions is fixed and all peers in all dimensions send data to their appropriate parent peer.

2. *Adaptive* - 'adaptive robustness': Peers are able to measure their surrounding churn behavior by analyzing successful and failed operations. Peers in the first dimension use this information to adaptively recruit parallel peers. For this, a minimum and maximum number of dimensions d_{min} and d_{max}, respectively, is defined. We propose a backoff algorithm where nodes react with a strong increase of the number of dimensions when churn behavior occurs and the number of dimensions degrades slowly in times of less churn to a value as low as d_{min}.

3. *Slotted Adaptive*: 'adaptive robustness + save bandwidth' - Starting with the adaptive mode, peers do not send their data to the appropriate parent peer directly. Instead, *successive slots* are created which define when each dimension should send its data to the corresponding parent peer. First slot is reserved for the peer in dimension one, second slot for the peer in dimension two, ..., $d_{current}$ slot for the peer in dimension $d_{current}$, with $d_{current}$ being the current number of dimensions the peer in the first dimension has chosen. A peer in dimension i sends its data if and only if the peer in dimension $i-1$ did not successfully send data up the tree for $i \in \{2, .., d_{current}\}$. The peer in dimension $i = 1$ starts at the beginning of each update interval.

In the evaluation, we implemented Mr.Tree atop SkyEye.KOM in the p2p simulator framework PeerfactSim.KOM. We compared the accuracy, the costs, and assess the cost-efficiency of our three approaches with four combinations in total and the vanilla protocol of SkyEye.KOM while varying node fluctuations. These are two combinations for the fixed mode with three and five dimensions ($F3$, $F5$), adaptive mode with $d_{min} = 3/d_{max} = 5$ and slotted adaptive with $d_{min} = 3/d_{max} = 5$. We define an approach to be cost-efficient if the accuracy gain is balanced with or even surpasses the costs. The outcome confirms that all four combinations outperform the vanilla protocol by far in terms of accuracy. The slotted mode proves to save bandwidth and provides the best cost-efficiency, even in low churn scenarios. The adaptive mode provides the best trade-off between accuracy and costs. It almost reaches the same accuracy as the fixed mode with d_{max} dimensions and always surpasses the fixed mode with d_{min}. We conclude that this mode is best suited for scenarios where higher node fluctuations are expected. Lastly the fixed mode delivers bad efficiency in scenarios with low churn, because of a high overhead owed to a lot of redundancy, but delivers the most stable accuracy due to no adaptive mechanisms. It is best suited for scenarios with high churn where smaller deviations are not tolerated.

2.2.4 Related Work on Enhancing Robustness of Tree-based Monitoring Solutions

In this section we address work in the field of tree-based monitoring systems which tackle the known issue of resilience. In the following, we pick up related work, which we elaborated in our paper, and add further work in this field. Jesus et al. [66] and Makhloufi et al. [79] gave first qualitative and quantitative evaluations, respectively, on tree-based monitoring systems which have issues with their robustness in environments with high churn due to single point of failures. In the latter paper, the evaluation shows that GAP [27] and SDIMS [131], which are classified in the same category, are prone to node fluctuations.

In Section 2.1 we present our tree-based monitoring approach extended SkyEye.KOM [45], which provides mitigation mechanisms to enhance its robustness. These mitigation mechanisms cover optimizations on the communication between peers and make use of structural characteristics. However, in environments of high churn these optimizations are not sufficient as the information flow is shut down. Therefore, with Mr.Tree, we propose a replication of both, structure and data, to boost the resilience.

The idea to replicate structures, such as trees, has been elaborated in the literature. The authors Li et al. [72] propose an aggregation and dissemination building block which is built bottom-up by using a parent function. Aggregate information flows bottom-up and the dissemination is performed in a top-down manner. Its main purpose is to broadcast information over the whole overlay by using the DHT. They tackle the problem of single point of failures by providing a multi-tree structure. For this, additional trees are announced by broadcasting the associated root in the existing tree, thus, peers eventually join the new tree. To increase the robustness of the system the authors propose to build up interior-distinct multiple trees. This can be achieved by choosing the parent function wisely. Mr.Tree is optimized for the use in monitoring services where the data availability has a high priority. Hence, we provide a transient storage which is used by peers at same position to synchronize each others. Another solution is presented by Bawa et al. with MultipleTree [10]. MultipleTree is a reactive approach in which peers build up k independent spanning trees. Here, peers wait till all active children pushed their data in all trees and forward their data up the tree. In short, whenever a query comes in, trees are constructed, data push is anticipated and forwarded to the parent and eventually the answer reaches the root. However, our approach follows a proactive pattern, therefore the trees need to get fixed, a data storage must be provided and long-term connections need to be maintained. The authors Artigas, García, Pedro and Skarmeta proposed an interesting approach called DECA [108]. DECA builds up an aggregation tree by using an underlying hierarchical DHT which is based on Cayley Graphs. The tree itself is set up by building subsets of the sets on the tree level above. These sets or tree nodes, respectively, consist of peers. Basically, peers gossip data in their leaf set and send this estimate up the tree by selecting a delegate in their parent cluster, which forward data further up the tree till it eventually reaches the root. However, due to the use of a probabilistic approach (gossip) it is expected to have inaccuracies. In Mr.Tree we focus on data replication and an exact data communication structure to improve the robustness while remaining on a high level of accuracy.

In the following we consider related work in other research fields such as content delivery in live streaming environments and sensor networks. Wang et al. propose mTreebone [121] which builds up a multicast tree to deliver live content in a top-down fashion. The tree-based backbone consists of stable peers, also mentioned as *super peers*, where mesh-based outskirts are attached. Another solution for live streaming is SplitStream proposed by Castro et al. [19]. SplitStream builds up forests of multicast trees by using Scribe [105]. These trees consist of interior-distinct nodes which lead to an enhanced reliability. Due to different requirements of live streaming applications and the problem of aggregating and disseminating of data, it is hard to compare these approaches. Mr.Tree uses a data maintenance to reliability collect and forward data whereas live streaming solutions follow principles on disseminating data as fast as possible while increasing delivery chance. So, the latter do not aim to reach a 100 % delivery ratio in high churn scenarios due to cost reasons. Furthermore, Mr.Tree supports homogeneous roles and does not build up super peer structures. Therefore, we assume that the fail probability of all nodes is equal. In the research field of sensor networks, peers aggregate data and send it towards a well-defined sink. Several solutions propose to use multipath routes to enhance the robustness of the routing. The big issue comes with duplicates which lead to a worsened precision, thus, they need to be filtered out. This can be put on a level with the aggregate of data in a tree by using multiple parents instead of just one. Considine et al. propose duplicate insensitive Sketches [23], which mitigate the accuracy issues using multipath routes. Mr.Tree uses logical distinct routes as each node and peer has only one parent, thus, problems with duplicates do not occur.

2.2.5 Personal Contribution

The author of this thesis, Andreas Disterhöft, had the initial idea on tackling the robustness problem of tree-based monitoring solutions by introducing logical nodes, so called hypernodes, which consist of several physical peers. Andreas Disterhöft and Phillip Sandkühler worked on a proof of concept and on the proof of work with fine-tuning on the protocol. During discussions the protocol design and the conclusively analysis were driven forward and Kalman Graffi was involved in some of the protocol decisions. Phillip Sandkühler is the main contributor of the implementation in PeerfactSim.KOM, whereas Andreas Disterhöft provided minor improvements which were important for the evaluation. Andreas Disterhöft was the person primarily responsible for the paper. He wrote the abstract, introduction, related work, evaluation and conclusion, whereas he and Kalman Graffi had discussions on the evaluation methodology. Phillip Sandkühler wrote the protocol section and additionally proofread and provided reviews of the paper. The last two points were performed by the other authors, Andreas Disterhöft, Andre Ippisch and Kalman Graffi, too.

2.2.6 Appendix Mr.Tree: Comparison and Additional Outcome

In this section, we extend the evaluation from our paper Mr.Tree [39] by the results of the extended version of SkyEye.KOM [45]. For this we broaden the evaluation table and added the extended (ext.) version for each churn scenario. Table 2.1 includes the approaches ext. and ext.$_{4*UI}$, whereas peers in the latter approach clean monitoring data in their local data base that is older than $4 * UI$. All other approaches use an interval of $2 * UI$. The $4 * UI$ interval is important to evaluate because a longer interval is needed by the mechanisms, so the data is most probably not deleted after a short period of time. Additionally, we added the evaluation of the exponential churn model with a one minute mean time.

Table 2.1 is row-wise divided into blocks of churn models, where the first block is the one providing the weakest churn behavior and the last block provides the strongest churn behavior. In each block, all Mr.Tree variants, the vanilla approach and the extended variants are shown. The columns include the three main metrics here, the averaged error of count, which is the metric for accuracy, the averaged costs per peer and coefficients compared to the vanilla approach. The accuracy contains the absolute error, the absolute standard deviation, relative error and its comparison to the vanilla approach together with the relative standard deviation and the comparison to the vanilla approach. The costs contain the absolute value in byte per second throughout the whole simulation, together with the overhead share and the comparison of the absolute costs to the vanilla approach. The last two columns contain the cost-precision coefficient, which is the multiplication of the accuracy and costs compared to the vanilla approach, and the cost-stability coefficient, which is the multiplication of the accuracy standard deviation and the costs compared to the vanilla approach. Therefore, the last two columns show the cost-efficiency of the approaches compared to the vanilla approach.

In the following we focus on the extended SkyEye.KOM approaches, their comparison to the approaches powered by Mr.Tree and the vanilla SkyEye.KOM approach. Then, we refer to the additional churn model with only one minute mean time and its outcome, which is an extra simulation scenario we did not add to the paper.

KAD churn model The first observation for the KAD churn model is that ext. can not significantly improve the precision compared to vanilla, while ext.$_{4*UI}$ can significantly improve its precision. This is due to the introduced mechanisms to exchange whole data sets of data and make them available where they are needed. By prolonging the clean interval we make sure that exchanged data sets will not be deleted in the next distribution interval of that peer who received these data. The adaption rate of the longer data set clean interval can be negatively influenced because old data is kept for a longer time span. However, the precision of ext.$_{4*UI}$ is still worse than the worst operating Mr.Tree mode, which is the slotted adaptive mode A3-5$_{slot}$. For the costs we state that both ext. versions cause higher absolute costs than A3-5$_{slot}$ but cause lower absolute costs than the other modes from Mr.Tree. This is mainly because the additional mechanisms exchange data bases between participating peers. On the one hand, this data base contains $bF+1$ aggregated data sets and is sent every time a peer changes its position in the tree. On the other hand, this data base contains just a single aggregated data set and is sent every time a peer has not sufficient information and therefore asks a sibling for its current view. Having a look on the second to last column, the cost-precision coefficient, we perceive that ext. is not efficient at all. Due to the same precision as vanilla and significantly increased costs, it reaches a value of 7. A coefficient below one means that the precision improvement covers the enhanced costs. The other extended mode ext.$_{4*UI}$ is as cost-efficient as adaptive

mode A3-5 and fixed mode with three dimensions (F3). In terms of the standard deviation of the precision, the extended approaches are not able to hold a stable precision like the Mr.Tree modes do. Therefore, the stability coefficient, which is the last column of the table, shows that the coefficient is worst for ext. and the second to last is ext.$_{4*UI}$. We conclude that ext.$_{4*UI}$ performs well in terms of the precision but is outperformed by the cost-efficient mode A3-5$_{slot}$. Its cost-precision coefficient is only better than the Mr.Tree mode F5, which uses five fixed dimensions and is therefore geared towards scenarios with high churn.

Exponential churn model with 30 – 5 minutes mean time Throughout all scenarios with increasing stronger churn behavior, the precision of both extended versions is worst. It is even worse than vanilla's precision while its costs are still the 2.5 – 3 fold. Therefore, its cost-precision coefficient is ranging from 3.4 – 4.6 for ext.$_{4*UI}$ and 5.8 – 8.6 for ext. Hence, it is never a better choice than the vanilla approach. This is due to a phenomenon we call the *starvation problem*. Peers operating high in the tree, that is, near the tree's root, do not receive any information of their children as they wait for valid data from their children. This is continued throughout the levels down the tree and due to this, peers are meant to starve because of the interrupted data flow. Eventually, this starvation leads to an erase of their data after $4 * UI$ or $2 * UI$, respectively. This effect is further amplified when decreasing the clean interval to $2 * UI$, which is reflected in the decreased precision. Additionally, the decreased precision is strengthen by the probabilistic approach of obtaining data from sibling peers and therefore assuming that their views are close to each others. We assumed the latter as the tree is meant to be balanced due to the hash function distributing peers evenly over the ID space. We conclude that, when it comes to stronger churn, the mechanisms of the extended versions of SkyEye.KOM are not able to improve the precision and the probabilistic approach plus the starvation problem decrease the precision compared to the vanilla version. Therefore, we advice against the usage of the extended versions in moderate to high churn scenarios.

Exponential churn model with one minute mean time The last churn model, which is a very tough stress test, shows some interesting points. It turns out that it leads to very strong stress for the overlay, indicators are the traffic overhead share which ranges from 10 % to 18 % for our Mr.Tree modes and a high failed operation rate (without figure). If the overlay cannot handle the churn, like here, the monitoring approach lying atop is doomed to degraded precision. The ranking shows that A3-5$_{slot}$ works better than F3 in terms of precision as the additional realities are needed here. Remarkable, the Mr.Tree modes still account a relative accuracy of 43.1 – 70 %, thus F5 performs only 7.6 percentage points worse than the vanilla approach did for Exponential churn with five minutes mean time. Vanilla's performance has broken down, its relative error is 83.9 % or, in other words, the precision amounts only 16.1 %, started with 91.2 % with KAD churn model. The precision of the extended versions is still worse than vanilla's with a relative error of 88.1 % and 93.4 %, respectively. A low standard deviation perceived in vanilla and extended versions is not helping out as their accuracy is worse in comparison to the Mr.Tree modes. The breakthrough point for the cost-precision coefficient, which is defined as the falling below one, is reached by all Mr.Tree modes, while the extended versions could not achieve it and range around 1.8. The superior cost-stability coefficient does not help vanilla and the extended versions because of their imprecise accuracy.

We conclude that the extended versions ext. and ext.$_{4*UI}$ are suitable for scenarios with low churn, such as the KAD churn model provide. But these approaches are always outperformed in terms of accuracy and costs by the cost-efficient slotted adaptive mode from Mr.Tree.

Table 2.1: Simulation outcome categorized by churn model. Scenarios ordered by absolute averaged error. Added extended SkyEye.KOM simulations to the table from [39].

Scenario	Averaged error of count						Averaged costs per peer			Coefficient (cf. Vanilla)	
	abs. ▲	abs. SD	rel.	rel. (cf. vanilla)	rel. SD	rel. SD (cf. vanilla)	abs. [byte/s]	overhead share	rel. (cf. vanilla)	error	SD
KAD churn model											
F5	55.7	21.9	0.014	0.155	0.006	0.056	1028.7	0.586	25.294	3.921	1.416
A3-5	58.6	45.3	0.014	0.164	0.012	0.116	483.7	0.587	11.893	1.950	1.380
F3	61.6	38.3	0.015	0.173	0.010	0.096	410.6	0.586	10.096	1.747	0.969
A3-5_{slot}	77.0	89.9	0.019	0.216	0.022	0.223	246.6	0.564	6.063	1.310	1.352
ext.$_{4*UI}$	100.2	143.2	0.025	0.284	0.036	0.357	279.8	0.806	6.880	1.954	2.456
ext.	334.1	313.2	0.085	0.963	0.080	0.806	296.6	0.817	7.292	7.022	5.877
vanilla	334.3	378.7	0.088	1.000	0.100	1.000	40.7	0.000	1.000	1.000	1.000
Exponential churn model, 30m mean time											
F5	56.3	80.3	0.012	0.178	0.016	0.222	1065.2	0.556	13.703	2.439	3.042
A3-5	60.5	101.4	0.012	0.192	0.021	0.282	526.0	0.536	6.766	1.299	1.908
F3	69.2	114.4	0.014	0.219	0.023	0.316	451.1	0.528	5.803	1.271	1.834
A3-5_{slot}	117.3	167.3	0.024	0.371	0.034	0.464	293.2	0.480	3.772	1.399	1.750
vanilla	314.5	359.7	0.065	1.000	0.074	1.000	77.7	0.000	1.000	1.000	1.000
ext.$_{4*UI}$	367.6	391.5	0.076	1.165	0.080	1.084	258.7	0.581	3.328	3.877	3.608
ext.	754.5	507.6	0.155	2.391	0.104	1.405	263.1	0.588	3.385	8.094	4.756
Exponential churn model, 20m mean time											
F5	65.0	82.0	0.013	0.158	0.017	0.196	1097.1	0.538	10.543	1.666	2.066
A3-5	80.0	131.9	0.017	0.195	0.027	0.317	577.3	0.508	5.548	1.082	1.759
F3	110.6	206.5	0.023	0.269	0.043	0.495	480.8	0.496	4.620	1.243	2.287
A3-5_{slot}	188.1	281.6	0.039	0.457	0.058	0.674	331.4	0.440	3.185	1.456	2.147
vanilla	408.7	415.3	0.085	1.000	0.086	1.000	104.1	0.000	1.000	1.000	1.000
ext.$_{4*UI}$	565.5	561.8	0.117	1.378	0.116	1.347	345.5	0.573	3.320	4.575	4.472
ext.	1022.0	527.2	0.212	2.489	0.109	1.265	357.8	0.588	3.438	8.557	4.349
Exponential churn model, 10m mean time											
F5	101.8	131.1	0.021	0.166	0.028	0.233	1188.7	0.493	6.458	1.072	1.505
A3-5	105.4	150.1	0.022	0.173	0.032	0.268	709.5	0.445	3.855	0.667	1.033
F3	110.3	174.2	0.023	0.181	0.037	0.311	567.7	0.421	3.085	0.558	0.959
A3-5_{slot}	211.6	236.9	0.045	0.347	0.050	0.423	440.5	0.360	2.393	0.830	1.012
vanilla	604.4	557.2	0.128	1.000	0.118	1.000	184.1	0.000	1.000	1.000	1.000
ext.$_{4*UI}$	811.7	516.0	0.171	1.333	0.109	0.918	569.0	0.545	3.092	4.122	2.838
ext.	1699.9	651.8	0.358	2.787	0.137	1.159	544.1	0.526	2.956	8.238	3.426
Exponential churn model, 5m mean time											
F5	157.0	210.3	0.034	0.148	0.045	0.291	1364.5	0.423	3.978	0.589	1.158
A3-5	168.2	267.1	0.036	0.159	0.057	0.371	927.4	0.364	2.704	0.430	1.003
F3	203.8	314.5	0.044	0.194	0.068	0.438	744.3	0.321	2.170	0.421	0.950
A3-5_{slot}	332.4	398.2	0.072	0.316	0.086	0.555	649.6	0.272	1.894	0.599	1.051
vanilla	1030.5	704.2	0.227	1.000	0.155	1.000	343.0	0.000	1.000	1.000	1.000
ext.$_{4*UI}$	1451.8	754.6	0.313	1.379	0.162	1.047	855.7	0.446	2.495	3.441	2.612
ext.	2532.0	595.3	0.544	2.402	0.128	0.825	834.3	0.434	2.432	5.842	2.006
Exponential churn model, 1m mean time											
F5	1162.7	679.7	0.303	0.361	0.175	2.313	2647.6	0.182	1.741	0.629	4.027
A3-5	1567.3	658.1	0.421	0.502	0.174	2.300	2313.0	0.138	1.521	0.764	3.498
A3-5_{slot}	1779.4	726.3	0.482	0.574	0.195	2.572	2114.3	0.101	1.390	0.798	3.575
F3	2031.3	619.1	0.569	0.678	0.170	2.249	2060.8	0.104	1.355	0.919	3.047
vanilla	2472.7	243.4	0.839	1.000	0.076	1.000	1520.7	0.000	1.000	1.000	1.000
ext.$_{4*ui}$	3558.0	328.8	0.881	1.050	0.078	1.035	2643.9	0.213	1.739	1.826	1.800
ext.	3775.2	150.4	0.934	1.113	0.034	0.455	2613.9	0.201	1.719	1.913	0.782

Chapter 3

Monitoring with Restrictions: Influence of Malicious Behavior

Beside problems concerning the robustness in tree-based monitoring systems, as stated in Section 2, we now refer to another open problem in such systems. That is, the presence of malicious nodes in these environments. We do not differentiate whether the malicious behavior is due to a physical malfunction, a bug in the protocol implementation or the node behaves malicious intentionally. Recent state-of-the-art monitoring systems assume that all nodes cooperate in order to comply with the protocol specifications. This is not necessarily true as stated in the surveys by Wallach [120] and by Urdaneta et al. [118].

In this chapter of the thesis, we investigate whether malicious nodes and their malicious behaviors, respectively, are able to interrupt, manipulate or even shut down the monitoring service running atop a p2p overlay. Furthermore and even worse, can nodes influence the monitoring service in such a way that the whole network will shut down by impacting the monitoring outcome?

Revealing an outcome: Yes, it is possible for malicious nodes to influence the monitoring results, and yes, this might have a huge impact on the whole network. This problem is particularly relevant for monitoring systems which create a common global view, that is, the data flow originates from aggregating data throughout all peers and, eventually, this view is disseminated to these peers. Furthermore, if data, extracted from the monitoring service, is used to adaptively react on the current p2p network's state, as proposed by Graffi et al. [48], it could have disastrous consequences for the quality of service. For example, imagine the actual hop count of a p2p network is quite high as the overlay perceives high churn. An infected monitoring service might show that the hop count is contradictorily low. Thus, the monitoring service will adapt the p2p overlay to the current condition by for example shrinking the routing table in its size or lower the number of known nodes per contact bucket. Ultimately, this would lead to an even higher hop count. This process is doomed to escalate and it can lead a p2p overlay breaking apart while nodes might get isolated.

Hence, we recapitulate that it is crucial to stress the importance of the data integrity and authenticity of the sender in monitoring services.

3.1 Convex Hull Watchdog: Mitigation of Malicious Nodes in Tree-Based P2P Monitoring Systems

This chapter contains the contribution, a summary and related work of our paper Convex Hull Watchdog [35], which is appended as a verbatim copy.

Andreas Disterhöft and Kalman Graffi. "Convex Hull Watchdog: Mitigation of Malicious Nodes in Tree-Based P2P Monitoring Systems". In: *Proceedings of the 41st IEEE International Conference on Local Computer Networks (LCN)*. 2016.
Acceptance Rate: 25.1 %.

After the preceding introduction, we state the contribution of our paper Convex Hull Watchdog [35] in Section 3.1.1 and proceed to the importance and impact on this paper on the thesis in Section 3.1.2. We give a summary of our paper in Section 3.1.3 and discuss related work in Section 3.1.4. We close with the declaration of the workload share in Section 3.1.5.

3.1.1 Contribution

Our first contribution is an attacker model, in which we categorize motivations attackers have and propose attacks for each category. Afterwards we propose DOMiNo, an approach to overcome and mitigate, respectively, malicious behavior of a tree-based monitoring systems' category. This approach is a two-part rating mechanism. The first part, the collecting rating, achieves a mitigation of data manipulation by utilizing a Z-Score-based rating, which is referred as the *convex hull*. The second part, the instant rating, is a collection of sanity checks to efficiently filter out implausible values and attacks to the structure. Whenever a malicious behavior has been detected, we reserve the right to drop data. For this we model the expected number of dropped data sets when doing so. DOMiNo is designed to passively rate the suspiciousness of incoming data by sniffing the monitoring traffic. By doing so, we prioritize accuracy over false positives and consume as few resources as possible. Evaluation shows that the impact of manipulation attacks and attacks to the structure can be mitigated or even filtered out, respectively, by applying our proposed Convex Hull Watchdog.

3.1.2 Importance and Impact on Thesis

We classify our paper as a solid work in the field of the second issue we investigate in the scope of this thesis, which is the influence of malicious behavior in tree-based monitoring overlays. As mentioned earlier, from a global point of view, we divide approaches dealing with malicious nodes into three components: the attacker model, the detecting mechanisms and the reaction upon detection. With our work we created a fundamental basis and proposed an attacker model, two detecting mechanisms, which are the instant and collecting rating mechanisms, and a simple yet effective solution for the reaction upon detection, which is a dropping policy. Future work can build on this basis by introducing new detection mechanisms and more sophisticated reactions. For example, nodes may try to rebuild, verify or even reconstruct the supposedly infected data.

3.1.3 Summary of the Paper

In our paper Convex Hull Watchdog we define an attacker model for malicious behavior in structured tree-based monitoring systems and propose our solution called DOMiNo, which includes the design decisions, its requirements and its core in the form of mitigation mechanisms.

For the proposed attacker model we first categorize malicious node's intentions into three categories. These are the general *manipulation* of monitoring results, the *free rider* behavior and the *disruptive* behavior. Nodes applying the manipulation behavior try to influence the monitoring view from other nodes. Here, we do not differentiate between nodes aiming on a defined target global view or not. Whereas, free riders try to maximize their benefit with as little own effort as possible. Lastly, disruptive nodes try to shut down the monitoring by not complying to the routing protocol or by eclipsing nodes. For the aforementioned categories we propose representative attacks.

We present three main design decisions which must be met for a solution. First, we aim for a *passive rating* mechanism, so, we do not add any communication overhead, instead, we operate on sniffed data only. Thus, no new attack vectors are added. Secondly, we prioritize high accuracy over a low false positive rate of the global view. Lastly, we aim to minimize computational costs.

Our solution, called DOMiNo, is divided into two parts: the detection of malicious behavior and the reaction upon detection. We introduce rating mechanisms, which are called watch dogs. These watch dogs are implemented to run in the local environment and rate suspicious behavior. On the one hand, collecting rating mechanisms mitigate the manipulation of data by using the modified Z-Score proposed by Iglewicz and Hoaglin [60], which is referred as the *convex hull* and its process is visualized in Figure 3.1. For this, new incoming data is checked whether it fits within the bounds of previous data by taking the statistical deviation into account. More specifically, all data from the previous measurement interval is taken, the median, the median absolute deviation (MAD) and the 3.5-fold of MAD is calculated, which is mentioned as our convex hull. A value outside of this convex hull is rated as suspicious and is eventually marked as malicious if repeatedly marked as suspicious. On the other hand, instant rating mechanisms include a couple of sanity checks for implausible values and can reveal attacks to the structure. The latter mechanisms double check whether the source ID conforms to the deterministic tree building required by DOMiNo. The requirements of DOMiNo are stated below. Whenever a data set is marked malicious, the node operating DOMiNo can react on this. We implemented a first reaction, which is a dropping policy. Also, we modeled the expected number of dropped data sets when such a dropping policy has been activated.

To be able to operate DOMiNo, we require the following. DOMiNo can run atop a structured monitoring solution which builds up a deterministic tree and pushes the aggregate proactively in a bottom-up fashion. These requirements are fulfilled by state-of-the-art systems like Cone [12], SDIMS [131], SOMO [137] and SkyEye.KOM [45].

For the evaluation we implemented DOMiNo atop SkyEye.KOM and ran several scenarios to obtain further insights. We ran a default scenario with no attackers and DOMiNo deactivated, a manipulation attack scenario with DOMiNo activated and deactivated and a scenario with all attack types activated. We summarize that we can mitigate manipulation attacks and limit them to the defined convex hull. Structural attacks can be filtered out efficiently and DOMiNo is able to prevent the counting to infinity-problem by detecting loop routing in the tree.

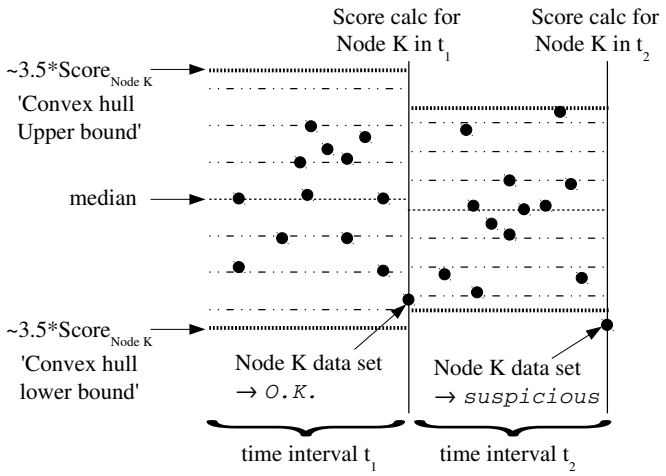


Figure 3.1: Schematic overview of the collecting rating mechanism. Each incoming data is inspected whether it fits within the convex hull boundaries. If it lies outside, it is marked as suspicious. This figure is taken from our paper Convex Hull Watchdog [35].

3.1.4 Related Work on Mitigation of Malicious Behavior

This section is based on literature discussed in the paper and we added few auxiliary statements. Related work in the field of secure decentralized monitoring is virtually non-existent. Current state-of-the-art structured monitoring approaches lack important security mechanisms. More specifically, we focus on the tree-based structured approaches collecting system-wide information (cf. Figure 2.2). Therefore, GAP [27], A-GAP [96], TCA-GAP [126], Cone [12], SDIMS [131], SOMO [137] and SkyEye.KOM [45] do not provide any mechanisms to counteract malicious behavior and their impact on the system. In the following, we examine security mechanisms in related research fields.

In the research field of wireless sensor networks (WSN) secure aggregation methods had been discussed. In the survey from Alzaid et al. [2], the authors state and categorize security mechanisms in the aggregation process of WSNs. The category most suitable for decentralized tree-based monitoring systems is the *multi-aggregator* model. In this model all inner nodes of the tree are aggregating data and forwarding the result to a common base station, which is equivalent to the root node of the monitoring tree. Two representative approaches are SDAP [132] and SHDA [20]. The authors of the former mechanism propose a divide-and-conquer together with a commit-and-attest scheme. The tree is divided into several sub-trees in order to reduce damage caused by malicious inner nodes. Furthermore, nodes commit their data and the root node can therefore check the correctness of the data. The latter approach follows an aggregate-commit-prove approach. The authors propose to append commitment to the aggregate and the base station broadcasts the aggregated view. Each node is then responsible to check whether its aggregate has been successfully added to the aggregated data. Potential drawbacks are the increased delay, the high communication and computation overhead. Approaches of secure WSNs can not be applied for our problem statement as they introduce new communication paths and a high overhead and therefore do no meet our requirement of being passive.

Reputation systems are proposed to rate the behavior of other nodes, for example the neighbors in a network. The survey by Marti and Garcia-Molina [81] list and discuss current reputation systems. The most famous representative is EigenTrust [67]. The authors propose a solution to distribute and manage so-called trust vectors describing the trust and transitive trust between participating nodes. In our solution we do not introduce any additional structures like a trust model, which itself can be attacked.

Intrusion detection systems (IDS) are commonly used in networks to supervise traffic and detect intruders or malicious behavior. The survey authored by Axelsson [6] discusses common IDS. These systems work passively by sniffing and rating actions. Generally, IDS work globally, need training to work reliably and are designed to work on large data sets to unfold its effectiveness. On the downside, our proposed solution operates on the node's local view and on small data sets.

In the research field of mobile ad-hoc networks watchdog mechanisms exist, like the solution proposed by Mogre et al. [83]. The authors observe and sniff the communication of neighbors. Unfortunately, such approach can not be adapted for p2p networks as the communication channels in p2p networks are directed by using unicast in the underlying network, the Internet, and communication paths are therefore hidden.

3.1.5 Personal Contribution

First, we refer to the workload share. Andreas Disterhöft, the author of this thesis, started this work together with Payman Alavi by defining first attacks in tree-based decentralized monitoring approaches such as SkyEye.KOM. In discussions fuzzy test cases and attack vectors were proposed and its evaluation analyzed. Payman Alavi implemented these attacks in the PeerfactSim.KOM simulator. Andreas Disterhöft refined the proposed attacks, added new attacks and categorized them, which were then implemented by Sebastian Brink. Andreas Disterhöft and Sebastian Brink discussed reasonable approaches in the research field of outlier detections and decided for the Z-Score-based approach which was finally implemented by Sebastian Brink. Andreas Disterhöft then implemented a reaction to supposedly malicious data, the optional drop of such data sets, and theoretically analyzed the number of dropped data sets. Furthermore, he introduced an evil bit counter in the data set and evaluated this metric together with new scenario setups for the paper's evaluation section. Kalman Graffi and Andreas Disterhöft had discussions about the protocol and the evaluation methodology. Now we refer to the workload share of the paper. In the following we state the main contributor for each section of the paper. Andreas Disterhöft wrote the introduction, except for the first paragraph, the related work section, the motivation of the attacker model, the introduction of the attack patterns, the blocking data sets paragraph, the evaluation and the conclusion. Sebastian Brink is mentioned in the acknowledgements as, unfortunately, after he finished his Master's thesis he was not interested in further steps of a paper but granted permission to use his work. He wrote the first paragraph of the introduction, the rest of attack patterns and the rest of the DOMiNo section. Sebastian Brink created Figure 1, Algorithm 1, Formulas 1 – 4, whereas Andreas Disterhöft made Figures 2 – 3, Formulas 5 – 6 and Tables 1 – 2. Kalman Graffi reviewed the paper and adapted parts of it.

Chapter 4

Monitoring with Restrictions: Partial Participation

In this chapter of the thesis we focus on another restriction in the monitoring of decentralized monitoring solutions, especially for tree-based monitoring approaches. So far, we introduced mechanisms for increasing the robustness when high node fluctuations occur and offer resilience in the presence of malicious nodes. In real-world environments another problem arises when applying tree-based monitoring systems in p2p applications. At this point we ask what happens if only a subset of all nodes participating in the p2p system also participate in the monitoring system. To give a short outlook: most state-of-the-art tree-based monitoring systems break down as they assume that all nodes participate in both, the p2p network and the monitoring system. In this section we detail this problem statement and present our proposed solution in the next sections.

The first leading question to motivate this field is the following: "Why should only a subset of all nodes participate in the monitoring?" We answer this question by referring to a typical scenario in the deployment of p2p software. Imagine an application is running on hosts distributed across the world and this application is built atop a p2p network. This network receives a new feature, for example the monitoring of such, via a software update. The salient point is that participating nodes gradually update their software at their own will. Thus, a share of all nodes stay with the old version for a certain time. To get a better understanding on the update behavior of users, the share of old versions and how long users stay with their outdated version, we reference to cross-sector scenarios. Having a look on the update behavior of the free runtime environment of Java [107], we observe that a lot of users do not update to the latest version. Only 50 % of all users updated to Java version 8, which has been released three years ago. We can observe a similar behavior for fee-based software like Microsoft Windows [85]. Returning to the use-case stated above, we conclude that this lazy update behavior results in a partly participation of all nodes in the monitoring service. This is critical as it is mandatory for the monitoring service to obtain the information about the whole network to report accurate data to services like the reacting mechanism proposed by Graffi et al. [48]. Hence, the question is how can we monitor the whole network while not all nodes are able to participate in the monitoring system because these nodes simply do not implement the monitoring protocol? The question is given in the next section.

The next leading question for the motivation is the following: "Why do tree-based monitoring systems suffer under partly participation?" This is due to the fact that such systems greatly

depend on an intact (tree) structure. Amongst others, examples are Cone [12], SDIMS [131], SOMO [137] and SkyEye.KOM [45]. If the underlying monitoring overlay is push-based for example, a node, which does not implement the monitoring feature, does not react on such data pushed to him. Most probably this node would simply ignore these messages. Therefore, this is equivalent to malicious behavior, where a node does not comply with the protocol and does not forward its monitoring data except that it does not ignore the specifications intentionally. This behavior leads to a disrupted structure and, eventually, to a vast loss of monitoring information. Hence, this results in a gradually reduction of the accuracy.

The third and last leading question is about the importance of the problem of a partial participation: "Why is it worth tackling the subject of partial participation?" In order to obtain meaningful results from the monitoring service, it must stay intact no matter whether all nodes or just a subset are participating in the monitoring process. If we are not able to capture the current state of the network, we are not able to react accordingly, e.g. counteract in times of poor network conditions like a high hop count. This leads to a decreased user experience due to a lower quality of service. Therefore, these monitoring results are badly needed by overlay optimization mechanisms to hold a certain quality of service.

In the next sections we present and discuss our solution called *Minicamp*.

4.1 Minicamp: Prototype for Partial Participation in Structured Peer-to-Peer Monitoring Protocols

This chapter contains the contribution and a summary of our paper Minicamp [37], which is appended as a verbatim copy.

Andreas Disterhöft and Kalman Graffi. "Minicamp: Prototype for Partial Participation in Structured Peer-to-Peer Monitoring Protocols". In: *Proceedings of the 42st IEEE International Conference on Local Computer Networks (LCN)*. 2017. Acceptance Rate: 28 %.

We start with the contribution of our paper in Section 4.1.1, followed by its importance and impact on this thesis in Section 4.1.2. A summary is given in Section 4.1.3 and we close with the declaration of the individual contributions in Section 4.1.4.

4.1.1 Contribution

In this paper we contribute a solution to overcome the problem of partial participation in current decentralized structured monitoring systems and give a short excerpt of its performance in the evaluation. We define active nodes to be nodes which support the monitoring feature and passive nodes to be those which do not support it. Our solution is called Minicamp and is designed to work as a middleware between the underlaying p2p overlay and the monitoring solution. We propose mechanisms needed by the middleware to fulfill the requirements of the monitoring solution, that are, the middleware must provide an overlay with active nodes and these nodes must hold monitoring information about both node types, active and passive nodes. We achieve the former by introducing methods to find active nodes in the p2p overlay and organize them in an additional overlay, the middleware overlay. We achieve the latter by proposing mechanisms to allocate passive to active nodes, which draw information from their assigned passive nodes and aggregate their data with the local data base. By doing so, we are able to operate a monitoring system run by active nodes which obtain a common global view by injecting monitoring information of both node types.

4.1.2 Importance and Impact on Thesis

The partial participation is the last problem we prosperously tackled in the research field of system-specific information monitoring. With our paper Minicamp we successfully attracted attention to the problem of partial participation in decentralized monitoring systems. The motivated problem statement as well as our proposed solution have been accepted by the community. This acceptance is a great achievement and therefore very important for this thesis. Furthermore, this paper has laid the foundation for an extended version, where we discussed further protocol details. This extended paper is presented in the following Section 4.2.

4.1.3 Paper Summary

The aim of Minicamp is to provide mechanisms to deal with the problem of partial participation in state-of-the-art tree-based monitoring solutions. These mechanisms should be universal and therefore Minicamp shall provide reusable structures for a broad number of tree-based monitoring approaches.

A schematic overview of the usage of Minicamp is shown in Figure 4.1 and described below. The Minicamp protocol is subdivided into three processes which run in parallel on *active nodes*. Active nodes are meant to be nodes that want to actively participate in the monitoring protocol, while passive nodes are not aware of the protocol. The processes are executed in a sequential order. These processes are the creating and maintaining of the middleware overlay, the probing to obtain monitoring data from passive nodes and the usage of a common monitoring overlay atop the middleware overlay in which the obtained data is injected. We detail these processes in the following. The first process is responsible for the discovery and organization of active nodes within one structure. In this paper, we do not discuss the discovery process but assume that such a mechanism is given. Active nodes create the middleware overlay or join it via bootstrapping a discovered active node. Next, active nodes divide the p2p overlay's ID space into disjunct ranges, in which they scan for passive nodes. This list of passive nodes is maintained by each active node and they are meant to be responsible for these passive nodes. Such scanning and maintaining mechanisms are assumed to be given and not discussed in the scope of this paper. However, this process is responsible to create and maintain a concise list of passive nodes an active node is responsible for. In the second process, active nodes probe their respective passive nodes. For this, they request the list of passive nodes from the first process and start to probe them to obtain monitoring information. An example probing method is the measurement of the passive nodes' bandwidth. This can be done using a method proposed by Goebel et al. [42] which utilizes ping messages being sent, the so-called packet trains, and interprets their responses. For each passive node the obtained data is stored locally in a data base. In the third and last process, a common monitoring overlay is applied atop the middleware overlay. The information drawn by process two is fed into this monitoring system. Thus, in addition to the local measurements, the data gathered from passive nodes is injected. By doing so, the monitoring system monitors metrics from all nodes. So, the common global view contains the active nodes' view and the passive nodes' probed view. In order to run Minicamp we state the following requirements. First, the p2p overlay should implement the KBR interface [25]. We require that the p2p overlay's ID space is one-dimensional, where both, nodes and objects, are located. Lastly, we need a convex node responsibility range to run Minicamp. In particular, this means that, if a node answers on ID_x, it is responsible for all IDs between its node ID and the ID_x. These requirements are mainly needed by process one to properly run its discovery and scan mechanisms.

At the end of our paper we evaluated Minicamp running a worst case scenario with an active node share of just 1 %, thus, 99 % of all nodes were passive ones. We identified the most important metrics to be the monitoring precision and the costs in terms of bandwidth and delay. For the monitoring precision we conclude that it can cope with the reference scenario, which is the scenario with 100 % active nodes participating in the p2p overlay, but suffers under the scan and probing error of process one and two, respectively. For the bandwidth costs we deduce that, due to the additional load of active nodes with a high share of passive nodes, weak nodes might get overloaded. In such cases we advise to promote support peers, which are peers one can delegate load to, as proposed by Graffi et al. [51]. To find suitable peers our

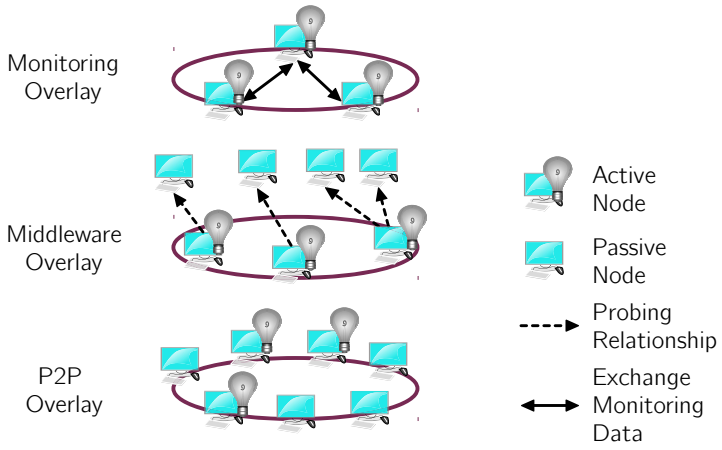


Figure 4.1: Schematic overview of Minicamp. Active and passive nodes are participating in the p2p overlay. Active nodes participate in the middleware overlay, passive nodes are mapped to them and they probe monitoring data. This data is injected in the monitoring overlay running atop.

solutions for the problem statement of peer-specific monitoring and the capacity-based search from Section 5 can be utilized. Lastly, the costs in terms of delay strongly depend on the used monitoring structure, active node share and the delay of the probing method. On the one hand, the delay might decrease when using a monitoring tree as only a subset of all nodes, more precisely, the active nodes, participate in such and the tree depth therefore shrinks. On the other hand, the probing delay negatively influences the monitoring delay.

4.1.4 Personal Contribution

Andreas Disterhöft, the author of this thesis, initiated the project with the goal of providing a middleware to be able to monitor the data of only a subset of all nodes in the overlay. More precisely, he proposed the first protocol ideas of creating a middleware, probe nodes and include the data into the monitoring system. Furthermore, he introduced the mapping of passive to active nodes and the responsibility of active nodes to obtain information from their mapped nodes. Together with Franz Kary algorithm details were discussed and the solution was implemented by Franz Kary. More precisely, he implemented three functionalities needed by Minicamp, which are stated in the paper under Step 1 in Section II A). These are mechanisms to find active nodes (via DHT, KBR), to organize these nodes in an active overlay and to scan and maintain passive nodes using different strategies (unilateral, bilateral and in general). Andreas Disterhöft created and implemented interfaces for the mechanisms of the paper's solution Step 1. He implemented the probing mechanism as proposed in Step 2 and created an interface for Step 3. He implemented this interface in SkyEye.KOM. For the evaluation, he motivated the probing method, performed the simulations and analyzed their outcomes. Kalman Graffi was involved in the whole process, discussed the protocol with Andreas Disterhöft and gave essential feedback on the evaluation methodology. The major work on the paper was performed by Andreas Disterhöft. He wrote the paper and took into account the reviews given by Kalman Graffi. Unfortunately, Franz Kary, who was involved into the preliminary work, had no time to work on an extension and the paper itself. Therefore, instead of adding him to the list of authors, we were forced to add him to the acknowledgements.

4.2 Minicamp: Middleware for Incomplete Participation in Structured Peer-to-Peer Monitoring Protocols

This chapter contains the contribution and a summary of our extended version of the paper Minicamp [36], which is appended as a verbatim copy.

Andreas Disterhöft and Kalman Graffi. "Minicamp: Middleware for Incomplete Participation in Structured Peer-to-Peer Monitoring Protocols". In: *Proceedings of the 32nd IEEE International Conference on Advanced Information Networking and Applications (AINA)*. 2018. Acceptance Rate: 28 %.

In the following we present the contribution of our paper in Section 4.2.1, whereas its importance and the impact on this thesis is discussed in Section 4.2.2. Further, we give a summary in Section 4.2.3 and declare the workload share in Section 4.2.4.

4.2.1 Contribution

In this paper we extend our contribution, stated in Section 4.1.1, by in-depth protocol descriptions and a detailed evaluation. More accurately, we propose mechanisms for the discovery of other active nodes in the p2p overlay. We detail the theoretical background and define the scan range needed for the p2p overlay to middleware overlay mapping. Furthermore, we offer pseudo code descriptions on how to scan and maintain the responsible area of active nodes. Strategies are depending on the responsibility management of the p2p overlay, therefore we divide such responsibility policies into three classes, namely bilateral, unilateral and the general case. For all pseudo code snippets we conduct complexity discussions. In the evaluation we extend the monitoring accuracy, discussed in the short version, by an analysis of the middleware overlay performance and costs with different active node shares.

4.2.2 Importance and Impact on Thesis

In our paper Minicamp [36], which we present in this section, we extended the paper Minicamp [37] by additional theoretical background and further protocol details. By doing so, we accomplished the publication of Minicamp. As already mentioned in Section 4.1.2, the accepted problem statement of partial participation in decentralized monitoring systems and its acceptance in the community is very important for this thesis. This paper extends this idea and has therefore a high impact on this thesis, too.

4.2.3 Paper Summary

In this paper we extend our publication of Minicamp with a consolidation of the theoretical background, its mechanisms and techniques.

First, we refer to the mechanism, which is responsible to discover active nodes in the p2p overlay. In particular, the discovery mechanism is utilized for bootstrapping, thus, to join the middleware overlay. We propose two ways to discovery active nodes, namely the KBR-supported and the DHT-supported discovery process. On the one hand, in the KBR-supported discovery, active nodes initiate lookup operations to distant IDs in the overlay and demand other active nodes to reply on receiving this message, whereas passive nodes simply forward this enveloped and therefore known message type. On the other hand, in the DHT-supported discovery, active nodes create and maintain a logically linked list, which is stored at well-defined places in the DHT. Active nodes discovering other active nodes simply access this list, which provides a fast random access, to obtain bootstrap nodes. When an active node has successfully joined the middleware overlay, it inserts its contact details into the list.

The second technique we deepen is the responsibility assignment of active nodes, where we map passive onto active nodes. For this, we exploit the ranges active nodes are responsible for in the middleware overlay, provide map functions to project the boundaries onto the p2p ID space and define the scan area out of it. Hence, each active node can locally determine its scan area it has to scan passive nodes in. We provide general function definitions for the projection between two overlays and the scan range each active node has to maintain.

In the scope of this paper, we provide algorithms for the scanning of the responsible area of an active node. We contribute a generic building block of pseudo code for initial scan and maintenance of the area. Further, we present techniques for general overlays and give optimized algorithms for *unilateral* and *bilateral* overlays. These overlays are defined according to the way the responsibility areas of nodes are specified. In unilateral overlays, such as Chord [115], a node is responsible for just one hand-side in the one-dimensional space till the neighbor. Instead, in bilateral overlays such as Pastry [104], a node is responsible for ranges on both hand-sides till the half-way to the next neighbor. The main idea of these scan algorithms is to initially charting the area by performing lookups, either in a divide and conquer way for the general case or by exploiting the overlay responsibility policy. This mapping of an area to a node is maintained by pinging the nodes and performing lookups on area border IDs. We provide estimated worst case runtime for charting and maintaining. In the following, we summarize the charting algorithms for the three identified cases. First, in the general case, we perform lookups for an ID in an uncharted area. Eventually a node answers which is responsible for the range starting from its node ID and ending at the looked up ID. Consequently, we add a new block to our charted area and refresh the uncharted area accordingly. In the unilateral case, we start to lookup the lowest or highest ID, respectively. A certain node answers and we can continue with a lookup for node ID +/- 1, respectively. By doing this we get a new node in each lookup iteration. The bilateral case is quite similar to the unilateral case but, instead of performing lookups in one direction only, we have to pay attention on both sides.

In the evaluation of our work we focus on performance metrics of the middleware overlay, like the completeness of scanning all passive nodes, the ID space coverage of active nodes and the correctness of it, the precision of the monitoring and the overhead generated by all Minicamp mechanisms. For the evaluation scenarios we vary the active node shares representing phases of publishing software. Further, the used protocol for the p2p and middleware overlay is Chord and we performed all scenarios under churn. We summarize the outcome as follows. The correctness of the scanned passive nodes in the middleware overlay reaches numbers above 90 %, having only 5 % false positives at maximum. The distribution of passive nodes over the actives is not exact equal as it depends on the node positions and used hash function of the

overlays. The bandwidth costs for 50 % active node share scenarios show a good distribution, whereas in 1 % active node scenarios peers need to perform around 20 lookups per minute for the passive node scan. A stable state is reached by scenarios with 10 % and higher active node share. Nodes start only six lookups a minute when entering the stable state. For the monitoring delay costs we support our findings of the first paper. Here, the active node share positively influences the freshness of monitoring data and negatively influences the node load.

4.2.4 Personal Contribution

This paper is an extension of the paper presented in Section 4.1.4 and therefore the preliminary work from Franz Kary also applies here. Andreas Disterhöft, who is the author of this thesis, provided more details of the proposed mechanisms and extended them with theoretical equations and pseudo code. In detail, all equations and algorithms were created by him. The requirements and the active node discovery are based on discussion between him and Franz Kary. The scanning complexity was estimated by Franz Kary and Andreas Disterhöft refined this model. Furthermore, Andreas Disterhöft simulated scenarios with an in-depth discussion about the middleware performance, the monitoring accuracy and the costs of Minicamp. He wrote the paper and received important feedback from Kalman Graffi.

Chapter 5

Monitoring Peer Capacities and Capacity-based Search

In this chapter we turn our attention to the second part in the scope of this thesis, the monitoring of peer-specific data. Here, instead of monitoring system-specific information in decentralized environments, as discussed in the previous sections, we introduce how to monitor participants' capacities through decentralized approaches, how to make them searchable and discuss the trade-offs in such mechanisms.

First, we define peer capacities. In general, we define a capacity being a combination of measurable performance attributes of participants in a network. We define single performance attributes as capacity spaces and a combination of capacity spaces is mentioned as a capacity. These are for example the available upload and download bandwidth, free main memory, available CPU cycles or free storage space. If the capacity is defined by the order above, a peer will have the capacity ($2\,\frac{MBit}{s}$, $40\,\frac{MBit}{s}$, $2\,\text{GHz}$, $40\,\text{MB}$). Our requirements for a solution are as follows. The search mechanism must be fast and its costs low. Furthermore, we require a *full-* and *k-* search. Given a search query we define the full-search as a procedure that outputs a set of all peers which meet the specified requirement. This query consists of CS ranges, with CS being the number of capacity spaces. An example for a full-search's query in a scenario with two capacity spaces looks as follows: <CPU: 500 – 2000, Upload Bandwidth: 5000 – 20000>. The units are known due to the context. Analogously, we define the k-search as a procedure that outputs a set of only k peers meeting the query's criteria. The query types for k-search and full-search do not differ. If the parameter k is greater or equals the number of hits the query can obtain, then k-search is equivalent to the full-search. On the one hand, the full-search is equivalent to the window query stated in the survey from Zhang et al. [134] and should return all peers which fulfill the requirements in all capacities spaces. On the other hand, the k-search should return the k nearest peers having at least the given minimal capacities. This is an intersection between a window query and a k nearest neighbor query in terms of the survey from Zhang et al. [134].

The main problem statement we tackle here is that peer capacities are highly heterogeneous but tasks in decentralized networks often do not consider this circumstance. Peers capacities' range from a relatively weak smart phone to high-end PCs but these peers have to fulfill the same jobs in vast use-cases. With the Internet of Things (IoT), this capability diversity is getting even broader, because the majority of these devices is quite limited in terms of their capacities. It is of high importance to differentiate between the monitoring of system-specific

and peer-specific data. System-specific information can be aggregatable, whereas peer-specific data can be hardly aggregated, because capacities must be assignable in order to search for peers meeting a given capacity. Solutions for the former do not fit the requirements for solutions of the latter if the capacities of all peers must be findable. Therefore, we require new or adapted solutions to search for peers fulfilling a certain capacity requirement.

Further use-cases for the capacity-based search are ubiquitous. In homogeneous overlays, such as Chord [115] or Pastry [104], peer roles are considered to be homogeneous although the peers' capacities are not. Thus, when running an indexing and capacity-based search solution in such scenarios, an overloaded peer may search for a stronger supporting peer to delegate the load and therefore enhancing the quality of service. In heterogeneous p2p overlays like Omicron [28] one may apply such capacity search service to search for strong peers, in terms of online time and bandwidth, to put them in important positions inside a logical node. Moreover it is of high interest to index the capabilities of peers in a continues manner. A peer, that was suitable for a certain position, role or job, can be ill-suited after only 30 minutes as mentioned by Oppenheimer et al. [86]. In the field of distributed computing, an indexing and search scheme to find a node capable to perform a certain job is of interest. Graffi et al. [51] illustrated this environment with clusters. Such a service can be implemented in a push-based manner, where a job holder pushes his jobs to suitable nodes. Even the research field of IoTs is qualified for capacity-based search. Guinard et al. [55] stress the need of resource discovery for nodes in IoTs.

The indexing structure must contain and maintain the capacity information of all participants in the p2p network. Furthermore, the number of dimensions of data should be dynamic, therefore we aim for a multidimensional structure. Zhang et al. [134] provide a survey about multidimensional indexing and search for p2p networks. However, we aim for a structure to handle highly dynamic data, it must handle churn and must be therefore reliable and the maintenance overhead should be low.

We tackled the problem statement, described above and in Section 1.1, in the following two papers. These papers, in particular CapSearch [34] and PacketSkip [32], are presented in the Sections 5.1 and 5.2, respectively. CapSearch provides a modified kd-tree of a fixed height by dividing the capacity dimensions into fixed classes. It therefore serves a solution for Variant 1 of the problem statement. PacketSkip, on the other hand, uses a modified Skip Graph underneath and maps the multidimensional capacity data on a one-dimensional space. It serves a solution suitable for the requirements of Variant 2 of this problem statement. Both services require to run atop a DHT overlay and they store their structures in it.

5.1 CapSearch: Capacity-Based Search in Highly Dynamic Peer-to-Peer Networks

This chapter contains the contribution, a summary and related work of our paper CapSearch [34], which is appended as a verbatim copy.

Andreas Disterhöft and Kalman Graffi. "CapSearch: Capacity-Based Search in Highly Dynamic Peer-to-Peer Networks". In: *Proceedings of the 31st IEEE International Conference on Advanced Information Networking and Applications (AINA)*. 2017.
Acceptance Rate: 29 %.

We start this section with the contribution of our paper in Section 5.1.1. After this, its importance and impact on this thesis is elaborated in Section 5.1.2 and a summary is given in Section 5.1.3. Literature in this field of research is then stated in Section 5.1.4. Lastly, we declare the personal contributions in Section 5.1.5.

5.1.1 Contribution

We introduce an approach called CapSearch which fulfills the demands stated in Problem Statement PS4 of Section 1.1 and describe its indexing and search schemes in detail. With CapSearch we propose an indexing structure, which works as a service atop a p2p network and supports fixed capacity dimensions with fixed classes. Two types of searches are proposed, the k- and full-search. When peers perform a k-search, they search for k contacts which fulfill the specified requirements. In contrast, peers performing a full-search, query all peers in specified capacity classes or ranges of capacity classes, respectively. For better understanding we provide pseudo code for the most important mechanisms of CapSearch. Lastly, we evaluate CapSearch by implementing and analyzing it in the simulator framework PeerfactSim.KOM. We analyze the performance and costs of CapSearch in well-considered scenarios and emphasize its strengths in terms of maintenance costs and k-search accuracy.

5.1.2 Importance and Impact on Thesis

In this section we present CapSearch, our first solution in the second field of this thesis, which is the field of indexing and search of highly dynamic and high dimensional data. Our first solution comes with a limitation, a preliminary fixed number of dimensions must be given. Our focus has been accepted by the community, which is to concentrate on an efficient indexing structure that ensures the search completeness of highly dynamic peer capacities. Also, this paper is the foundation for further work, like the paper PacketSkip [32] we present in Section 5.2. Therefore, we conclude, that this paper is one of the most important papers in this thesis.

5.1.3 Paper Summary

The properties and characteristics of our solution CapSearch, which tackles the Problem Statement PS4, are declared as follows. We fixed the number of dimensions or capacity spaces, respectively, to be taken into account. So, for example we might limit the number of dimensions to three, which are, the available CPU cycles, free storage and available download bandwidth. Furthermore, the number of capacity classes is fixed as-well and covers the whole capacity space. However, the number of capacity classes can vary for each capacity space. We extend the example from above and the three capacity spaces CPU, storage and bandwidth could be divided into:

- Four CPU classes: 0.0 – 0.5 GHz, 0.5 – 1.0 GHz, 1.0 – 2.0 GHz, 2.0 – ∞ GHz.
- Six storage classes: 0.0 – 50 KB, 50 KB – 50 MB, 50 MB – 2 GB, 2 GB – 5 GB, 5 GB – 20 GB, 20 GB – ∞ GB.
- Five download bandwidth classes: 0.0 – 16 Kbit/s, 16 – 50 Kbit/s, 50 – 100 Kbit/s, 100 – 500 Kbit/s, 500 – ∞ Kbit/s.

In the following we sketch our solution, of which important mechanisms are shown in Figure 5.1. CapSearch uses an indexing approach to proactively arrange the peers in specific classes. It uses a modified binary kd-tree. The structure is used to divide and organize the fixed multidimensional capacity space. This is done by dividing one after the other capacity space per tree level into exactly two spaces. The creation of the kd-tree is deterministic as the division is performed in a particular and well-defined order. The division finishes, if the space can not be divided any more, thus, the leaves of the tree represent a single capacity class combination. In Figure 5.1 we show an example with four leaf nodes holding the permutations of CPU $= \{1, 2\}$ and MEM $= \{1, 2\}$. Each leaf node stores contact information from participants whose capacities are currently in the representing capacity. On the other hand, inner nodes store a snapshot of their induced sub-tree. This snapshot contains a detailed or an aggregated view on its sub-tree leaf nodes' IDs. In short, inner nodes provide views which are equivalent to leaf node references that store at least one participant in the capacity range represented by this inner node. We store the complete kd-tree in the underlying p2p network's DHT. The nodes of the kd-tree are logical nodes and are thus stored as objects at well-defined places in the DHT. Each peer is able to access a node by performing a single DHT lookup, of which complexity typically lies in $\mathcal{O}(\log(N))$. Peers responsible for logical nodes feel responsible to overtake functionalities to maintain the tree structure. The maintenance consists of three mechanisms: First, each participant of the p2p network periodically announces its capacity to the appropriate leaf node. Second, peers responsible for leaf nodes periodically clean outdated peer information which, for example, left the network without notifying the node. Furthermore, whenever a leaf node has been created, and therefore holds more than or exact one contact information, or a leaf node has been deleted, as the last contact information has been removed, a notification is sent up to its parent node. Third, peers responsible for inner nodes update their view of the sub-tree upon receiving a notification from their children and send this notification up the tree till this notification reaches the root. By doing this, the structure and the views of the inner nodes up to the root can update accordingly. We summarize the maintenance complexity as follows. The publishing of data only takes one DHT lookup. In the worst case, this publishing can lead to subsequent notifications up the tree and

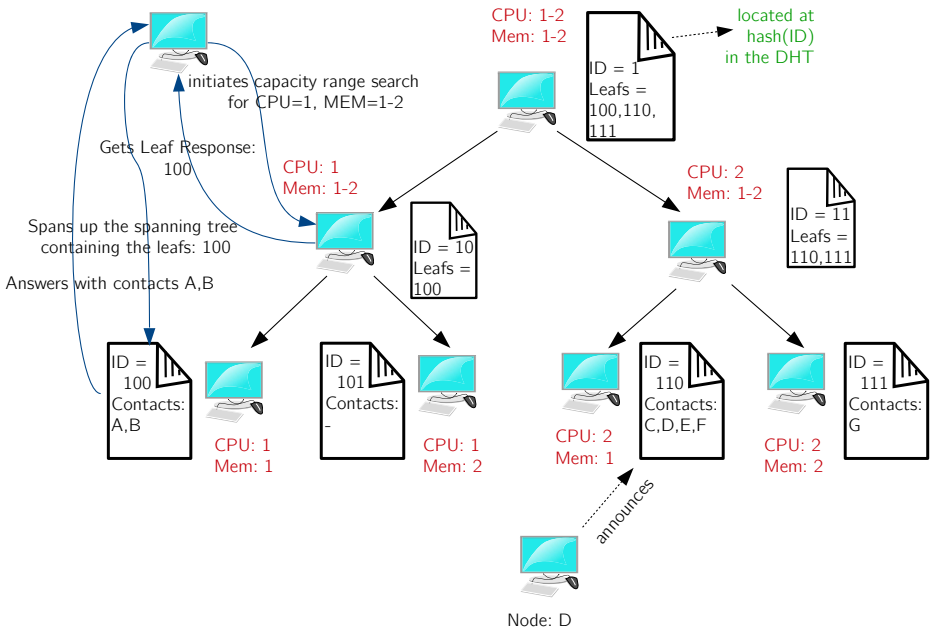


Figure 5.1: Overview of our solution's topology, its maintenance mechanisms and the search. This kd-tree holds two capacity spaces (CPU and MEM) with two capacity classes each. On the tree's first level CPU is divided and on the second level MEM is divided, which is the leaf node level. The documents beside the computers symbolize the DHT entry for this tree node. The computer on the left top had initiated a search, which has been answered successfully. This figure is taken from our paper CapSearch [34].

therefore takes the height of the kd-tree lookups in sum. This leads to exactly $\sum_{i=1}^{n} \lceil \log_2 k_i \rceil$ lookups, where n is the number of capacity spaces and k_i the number of capacity classes of capacity space i.

In the following, we refer to the search mechanisms of CapSearch. The search mechanisms for k- and full-search are quite similar. We refer to Figure 5.1, which shows an example for a search. The peer on the top left initiates a search with the query: CPU $= 1$ and MEM $= 1 - 2$. Given this range, firstly, the common parent for this capacity range(s) is calculated, so that, it overlaps the whole capacity range to query. Secondly, a lookup is performed to obtain the view of this inner node, which is a list of leaf IDs holding potential relevant participants. For the k-search, this list is sorted by distance to the lower bound of the query and this list is traversed by querying around or slightly more than k leaf nodes in parallel. For the full-search, this list is traversed by spanning up a binary search tree where the leaf nodes or the peers responsible for these nodes, respectively, span up this tree and report hits to the requesting peer. In our example this leaf ID list contains only one entry (100), which is obtained directly from the according leaf node. In the worst case of the k-search, given the requirement that all views of inner nodes are correct, the complexity in terms of hops needed to fulfill the operation is $\mathcal{O}(k * \log N)$, as $1 + k$ lookups are needed to obtain the contact information of peers that are located in pairwise distinct capacity class combinations. For the full-search, the complexity

increases to $\mathcal{O}(N * \log N)$, as $1 + N$ lookups must be performed, if the search range queries for all participating peers in the network and all peers are located in pairwise distinct capacity class combinations. For the time complexity, k-search can be completed in $\mathcal{O}(\log N)$, as, after obtaining the leaf nodes with the first lookup, the remaining k lookups can be performed in parallel. Full-search's time complexity is $\mathcal{O}(\log^2 N)$, as the binary search tree containing N nodes must be fully traversed, which is done in parallel. Thus, the height of the tree, which is $\log N$ in the worst case, multiplied by the time needed for a DHT-lookup ($\log N$) is required.

In the evaluation, we implemented CapSearch in our p2p simulation framework Peerfact-Sim.KOM und let peers join a p2p overlay (Chord), periodically change their capacities and initiate search queries for certain capacities. We analyze the impact of four scenarios: with churn disabled, with churn enabled, rising the number of capacity spaces and shorten the interval each peer adjusts its capacities in order to stress the maintenance. The study shows that the load induced by the maintenance of the kd-tree structure is well distributed amongst almost all participants. Therefore, each peer individually faces a low maintenance overhead. For the k-search, we experience a high precision (> 90 %), even when performing under churn or having a deeper kd-tree as a result of rising the number of capacity spaces. For the full-search, we see a high but degrading precision when operating under churn. Still, the precision amounts 58 % at minimum with a very low false positive rate (< 1 %).

5.1.4 Related Work

When studying the field of capacity-based search, several research fields, which are related to the one discussed here, can be considered. In the following we focus first on solutions operating on the same field and refer then to the more general major area of research for multidimensional indexing and search. Our investigation is based on the related work discussed in the paper and is expanded by further consideration on additionally relevant papers.

Graffi et al. propose SkyEye.KOM [51, 50], of which system-specific monitoring mechanisms we detailed in Section 2.1, a solution enabling a capacity-based search using the monitoring tree structure. For this purpose, tree nodes locally store a list of peer capacities from peers participating in their sub-tree. As peer capacities are not aggregateable, this list linearly grows with the tree level. In order to not overload inner nodes closer to the root node, this list is limited by a certain threshold. Therefore, the search mechanism does not take all peers' capacities into account, thus, the search results are incomplete and only k-search is supported. In our solution, we do not discard any capacities in the proposed indexing scheme. Hence, the search mechanism is able to obtain all capacities in its output. Additionally, our solution supports another search mode, the full-search.

In another paper in this field the authors Wette and Graffi propose adaptive heterogeneous AH-Chord [124]. In AH-Chord several Chord (ring) instances are maintained, one instance per capacity space, to index its participants' capacities. Here, peers calculate their ID depending on their current available capacity. By doing so, participants can search for a capacity by performing a lookup in the particular Chord instance. Range searches are realized by traversing nodes between the given lower and upper bound. However, maintaining one Chord instance per capacity space is expensive. Additionally, high data dynamic, that is the peers' capacities, leads to strong churn which needs to be reliably handled by Chord. With Fractal [68] the

authors Kargar and Mohammad-Khanli propose a Chord-like structure containing super-peers to provide a resource discovery service in IoTs. Those super-peers act as gateways to other Chord instances. Node dynamics may distract the structure and, especially, leaving super-peers may lead to isolated Chord instances. In contrast, in CapSearch, we propose the usage of a kd-tree structure to handle multidimensional data and DHT replication of the underlaying KBR-compliant p2p overlay is utilized to provide churn-resilience.

In the major area of research for multidimensional indexing and search there are several interesting and related approaches to our proposed work CapSearch. In a survey paper by Zhang et al. [134], the authors categorize these solutions and provide an evaluation map of p2p-based multidimensional index (cf. Figure 2 in [134]). In CapSearch we use a kd-tree structure, thus, we focus on related work using kd-trees, space filling curves and binary search trees here. In the following, we refer to work utilizing a kd-tree to partition the multidimensional space like we do with CapSearch.

Gao and Steenkiste propose DKDT [41], a Chord-based and DHT-supported approach, where kd-tree nodes are mapped onto the DHT by using a hash function. Each tree node maintains its local database which contains limited information about the tree structure. This Tree Information Base (TIB) is kept current by periodically probe nodes. It is highly needed as the tree structure is dynamic and nodes split up or merge, respectively, when reaching a certain threshold. The search mechanism utilizes the TIB to forward the query to find either the nearest neighbor or the k nearest neighbors. A similar work to DKDT is proposed by the authors Tang et al. [116]. In m-LIGHT a method is proposed to preserve the data locality by smartly generate DHT keys. Furthermore, the data-aware splitting strategy comes into play to load balancing the leaf nodes of the kd-tree. In our work CapSearch, we focus on indexing dynamic and ever-changing data, which could lead to frequent split and merge operations, respectively, in m-LIGHT. Furthermore, we keep the space partitioning simple and require a static kd-tree space partitioning. Furthermore we use the DHT-support for a reactive maintenance of the structure and process range queries not step by step by traversing inner nodes of the kd-tree, instead, we request one inner node and then traverse the needed leaf nodes to obtain the data.

MURK [40], an approach using the CAN [100] overlay for the topology and a kd-tree for the space partitioning, is proposed by Ganesan et al. Depending on the peer load the peers partition their space and use a greedy routing based on the routing in CAN. To overcome the poor routing complexity from CAN, the authors use Skip Lists to create shortcuts to distant locations. Another approach using a kd-tree for space partitioning and a Skip Graph for maintaining the data, which contains Skip Lists, is SkipIndex [133] and proposed by the authors Zhang, Krishnamurthy and Wang. Here, each kd-tree node is identified by an ID representing its split history. The Skip Graph stores the kd-tree's leaf nodes, which contain the data. By doing so, the leaves' order is preserved. Search is performed by scanning the local kd-tree view and forwarding to regions by using the Skip Graph. In contrast, we focus on a DHT-supported structure for the routing and a modified kd-tree for fast search. Here we access in one lookup all regions which have to be traversed to obtain the data needed.

Lastly, we refer to approaches using a binary tree and space-filling curves to map the multidimensional space onto a one-dimensional space. Jagadish et al. propose BATON [62], its extension BATON* [63] and a variation called VBI-tree [64]. A balanced binary tree is used for the indexing structure, where each peer of the p2p overlay maintains one node in the tree.

Each node is responsible for a range and has certain connections to other nodes, such as its brother, cousin, parent and grandparent. BATON supports range search for one-dimensional data by traversing this tree structure. BATON* supports multi-attribute range search by using space-filling curves to map it on the one-dimensional space by dividing it into regions and route along this tree of regions. VBI-tree is a variation of BATON to support real multidimensional queries by shifting data to the leaf nodes. Carlini et al. propose Dragon [18], a similar approach which uses a binary tree and space-filling curves. Here, inner nodes, which are hosted by peers in the p2p overlay, maintain digest information of their sub-trees and leaves maintain multi-attribute data. multidimensional data is mapped on the one-dimensional binary aggregation tree by using a space-filling curve. The search is performed bottom-up and is therefore started at leaf level. Our solution CapSearch uses a kd-tree instead of a binary tree and operates on mapped data. The routing in BATON* might be inefficient if routing tables are outdated due to peer dynamics. Furthermore, we prioritize to handle dynamic data and search is performed top-down by asking one inner node first and then visiting particular leaf nodes.

5.1.5 Personal Contribution

Andreas Disterhöft, who is the author of this thesis, started the work on this project with the aim to divide the fixed number of capacities in the multidimensional capacity space into fixed classes and map nodes, with their respective capacities, onto this space. Andreas Disterhöft and Ivan Tolstun discussed appropriate data structures and Ivan Tolstun chose to start in his Bachelor's thesis with kd-trees to split the multidimensional capacity space. The ID structure and the storage of user's capacities in the leaves were driven forward in these discussions as well as its refinements; the access of data, its maintenance and the search for capacities. Ivan Tolstun implemented a first prototype of CapSearch and analyzed the outcome with Andreas Disterhöft. Afterwards, Andreas Disterhöft proposed an optimized version based on the outcomes of this work. He reimplemented and reconstructed the prototype and introduced new algorithms. In particular, the kd-tree's data structure received a rework as the prototype structure's maintenance was too expensive and the search too time intensive. Due to the new data structure, Andreas Disterhöft implemented a new search algorithm and subdivided it into the k- and full-search. Furthermore, he shifted the transient kd-tree structure, which was using pings for the maintenance, to a persistent structure with DHT support. After further bug fixes, he re-evaluated the new approach with a new set of metrics and scenarios. Kalman Graffi, the other author of this paper, contributed with discussions about the ideas, the protocol, its enhancements and the methodology of the evaluation. The paper presented in this section was written by Andreas Disterhöft. Kalman Graffi gave valuable reviews which were taken into account by Andreas Disterhöft. Ivan Tolstun was involved in the first prototype of CapSearch, but he did not participate in the paper as the first prototype was inefficient and needed a rework. Unfortunately, Ivan Tolstun was not interested in proceeding work on the prototype, therefore we gave him credit in the acknowledgement section of the paper for his preliminary work.

5.2 PacketSkip: Skip Graph for Multidimensional Search in Structured Peer-to-Peer Systems

This chapter contains the contribution, a summary and related work of our paper PacketSkip [32], which is appended as a verbatim copy.

Andreas Disterhöft, Andreas Funke and Kalman Graffi. "PacketSkip: Skip Graph for Multidimensional Search in Structured Peer-to-Peer Systems". In: *Proceedings of the 11th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*. 2017. Acceptance Rate: 20 %.

We open this section with our contribution to the paper in Section 5.2.1, which is followed by its importance and impact on this thesis in Section 5.2.2. We present a summary of PacketSkip in Section 5.2.3, whereas its related work is given in Section 5.2.4. We close with the personal contributions of this paper in Section 5.2.5.

5.2.1 Contribution

We contribute with a proposal of a further solution for the indexing and search of peer capacities problem we stated in Problem Statement PS4. We aim for a more flexible solution than CapSearch [34], so that the number of capacity spaces and capacity classes can adapt itself during runtime. It is therefore compliant to Variant 2. In our paper we propose PacketSkip, an indexing and search service for highly dynamic and high dimensional data, like peer capacities, in decentralized networks. With PacketSkip we present a solution that uses a Skip Graph structure under the hood and we propose to move the peer-centric structure of Skip Graphs towards a data-centric structure. We detail how PacketSkip, which runs atop a KBR-compliant p2p overlay, implements the full- and k-search mechanisms. Lastly, we implement PacketSkip in our simulator framework PeerfactSim.KOM and evaluate it. Compared to CapSearch, we verify a superior near perfect search accuracy and a fast adapting maintenance while introducing a small overhead which is unequally distributed among the peers.

5.2.2 Importance and Impact on Thesis

In this section we present our second solution in the second field of this thesis. Here, we propose further improvements in this field that have been accepted by the community. These improvements are the adjustable dimensionality of the data and the increased accuracy of the full-search mechanism. It was of high interest to overcome the limitation of CapSearch regarding the fixed number of dimensions, thus, this paper is as important as CapSearch is for this thesis. As the choice between CapSearch and PacketSkip depends on the use-case, both papers might have an impact in the literature. If the use-case can fix the number of capacities and its classes and the k-search is mostly used, CapSearch is the right choice as it has less maintenance costs. Furthermore, if both search mechanisms are used and the number of capacity spaces and classes is small, then CapSearch might be a better choice. Otherwise, if the use-case postulate a variable (and even high) number of capacities and an accurate full-search, PacketSkip is the preferred solution.

5.2.3 Paper Summary

In this section we summarize our paper PacketSkip [32] starting with the main properties of this solution, continuing with a sketch of PacketSkip with its maintenance, index and search mechanisms and closing with outcomes from its evaluation. Just like CapSearch [34], which we presented in Section 5.1, we present a solution for the problem statement of indexing peers' capacities and search for them. In PacketSkip the number of dimensions or capacity spaces, respectively, is dynamic at runtime. Thus, in contrast to CapSearch, the number of capacity spaces does not have to be set at the beginning of the service. So, participants are able to add more capacity spaces during runtime to fit certain application's needs which may change during runtime. Furthermore, the number of capacity classes is unrestricted and can be chosen from $\mathbb{N}$. These two properties combined mean that capacity spaces can be added at will and they are treated as labeled numbers from $\mathbb{R}$ and therefore not categorized in static spaces and classes as presented in CapSearch.

The main idea of PacketSkip is to store multidimensional peers' capacities in a single one-dimensional sorted space by storing each dimension's value together with its label. This label is mentioned as the capacity feature and may be, for example, the bandwidth (BW) or the available CPU cycles (CPU). We utilize a Skip Graph [5] for such structure to provide a one-dimensional sorted space. In PacketSkip, Skip Graph nodes are stored as DHT objects under their Skip Graph node ID (hash('node ID')). An example of the PacketSkip's topology storing two-dimensional data (BW and CPU) in the DHT is shown in Figure 5.2. Each Skip Graph object stores several peers' capacities, in the following mentioned as elements. This number of elements is capped with l. The maintenance mechanisms used to maintain PacketSkip's Skip Graph are reactive. The initial node is created by a peer which is responsible for a well-defined ID in the DHT. This node is therefore responsible for the whole sorted space ($\mathbb{R}$). When reaching the number of elements per object l, this node either performs load balancing mechanisms or creates a new node and splits its space into two parts, so that each node maintains an equal number of elements. Thus, the number of objects grows with the number of peers. When the number of elements shrinks and the affected node's neighbors can overtake the remaining elements, a node can be deleted by shifting the responsibilities to its neighboring nodes. Before creating new nodes or deleting nodes with a small number of elements, nodes or their responsible peers, respectively, try to even the number of elements across their direct neighbors by shifting the responsibility range of their nodes and move corresponding elements accordingly. By doing this, the load is balanced throughout the nodes and neither unnecessary new nodes are created nor required nodes are deleted. As not each peer in the p2p overlay is responsible for a PacketSkip node, we propose a mechanism to find a node or bootstrap into the system, respectively. Each peer responsible for a node announces its Skip Graph nodes' IDs in a well-defined ring buffer in the DHT. By doing so, other nodes can access this ring buffer to obtain a Skip Graph node ID and access the Skip Graph node by performing a DHT-lookup for this ID.

In the following, we describe the mechanism to store and update peer capacities. We propose a proactive approach where each peer's capacities are stored on *CS* places, with *CS* being the number of capacity spaces, along with the other *CS-1* capacity spaces. For example, a peer wants to announce its three capacity spaces being CPU, BWD and BWU. The values for the spaces are as follows: 2,000 (CPU), 50,000 (BWD) and 5,000 (BWU). In the PacketSkip node, which is responsible for the value 2,000, the labeled value 2,000 CPU together with the other spaces (50,000 BWD and 5,000 BWU) are stored. Analogously, elements are created in

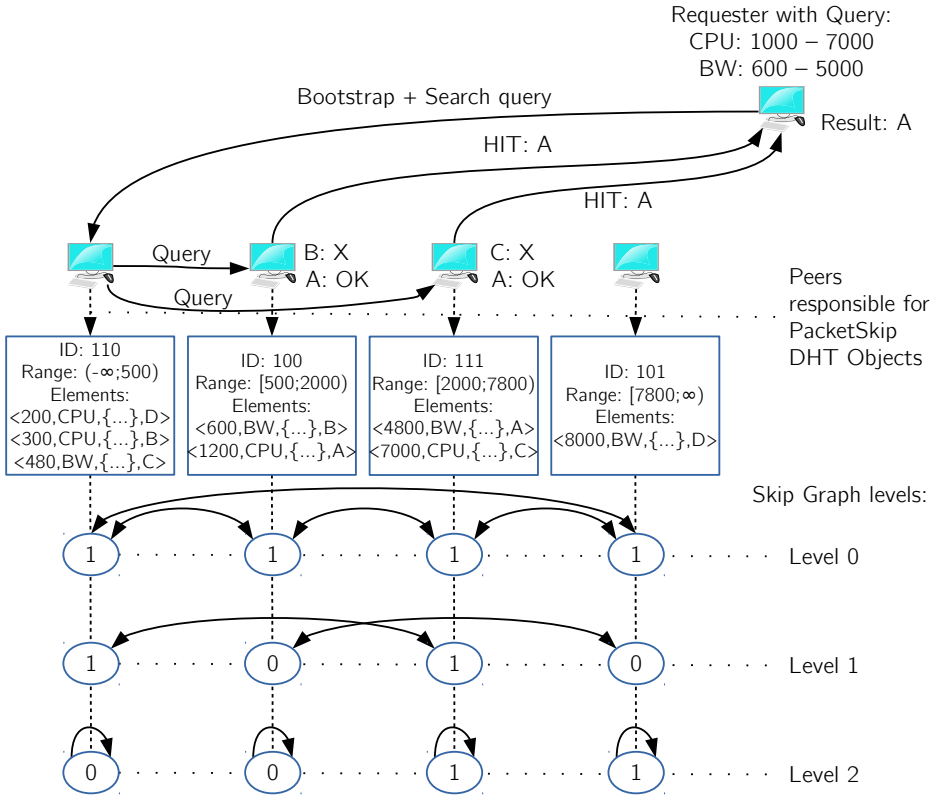


Figure 5.2: Overview of the topology of PacketSkip and example of a full-search query. Two-dimensional data (CPU and BW) from several peers are stored in four PacketSkip nodes. The lower part of the figure shows the Skip Graph levels, the boxes represent PacketSkip nodes stored in the DHT and the upper part visualizes the search procedure.

PacketSkip nodes which are responsible for 5,000 and 50,000. We propose to store the other capacity spaces as additional capacity features as-well to speed up the search mechanisms. By doing so, search hits are evaluated during the search process by the responsible peer and not by the search initiator. In the following we detail the process of capacity updates. A peer first bootstraps into the PacketSkip, hands over a message containing the insert and, if required, a delete query for the old capacity. The query is then greedily forwarded to near contacts by sub-dividing the query range. By doing so, a logical search tree is created and nodes receiving such a query store appropriate values, if it falls into the range of the node or proceed the search by further sub-dividing the space and forwarding the query to near regions. Eventually, all values are stored and the update operation finishes. In environments with churn it is mandatory to truncate outdated elements, therefore, peers, which are responsible for PacketSkip nodes, delete outdated elements after a certain time interval. These elements are therefore only deleted, if they have not been updated by the respective peers. The hop complexity of the capacity update process in a network of N participants is the following: $\log(N) + 2 * CS * \log(\frac{N}{l}) * \log(N)$ in the worst-case, which can be simplified in the $\mathcal{O}$-notation to $\mathcal{O}(CS * \log^2(N))$, whereas CS is usually a small number.

Next, we split pars of the term and go into more detail:

- '$\log(N)+$': Finding a bootstrap node to PacketSkip's Skip Graph data structure.
- '$2*$': A capacity update includes an insertion of the new capacity and a deletion of the old capacity. The deletion is not mandatory due to the soft-state approach but decreases the number of false positives in search queries.
- '$CS*$': Each of the CS capacity spaces must be updated and their locations in the Skip Graph might be far apart. For example 700 (MB) and 50,000 (BWD). Queries are forwarded in a temporary search tree in CS different directions at maximum.
- '$\log(\frac{N}{l})*$': This represents the number of hops needed to access the target PacketSkip node. This equals the routing complexity in Skip Graphs.
- '$\log(N)$': This represents the DHT lookup complexity, which is needed to access a single PacketSkip node

For the time complexity, we adapt the above term in regard to the parallel running queries. The insertion and deletion and the queries forwarded down the search tree are processed in parallel, thus, for the time complexity we expect a time complexity of $\log(N)+\log(\frac{N}{l})*\log(N)$. This can be simplified to a worst-case runtime in $\mathcal{O}(\log^2(N))$.

After having discussed the main idea, the maintenance and capacity update mechanisms discussed, we now refer to the search mechanisms in PacketSkip. For this, we refer to the example from Figure 5.2. Here, the requester on the top right of the figure performs a full-search for CPU: 1000 – 7000 and BW: 600 – 5000. First, the query initiator checks whether the local peer contains a PacketSkip node. If not, a lookup for such node is performed by following the bootstrapping steps stated above. Then, the query is passed to the obtained node and the idea is to traverse all values for the dimension with the shortest range, which is BW in this example ($5000 - 600 < 7000 - 1000$). In general, this range is traversed with a divide and conquer approach by forwarding chunks of this range to known contacts in this region. Therefore, the range is traversed by creating a search tree, similar to the one proposed for the capacity update mechanism. On receiving such a query, the responsible peer for the PacketSkip node reports a hit to the initiator containing the peer-contact, if all(!) dimensions fit the desired range, thus, the reported peer fulfills the search query. Furthermore, the peer continues and sends the query to the next peer down the search tree, if it is not responsible for the whole chunked range. This equals of not being the leaf node of the search tree. Eventually all nodes of the search tree are traversed and the search operation finishes. Applying the procedure on our example in Figure 5.2, after bootstrapping the PacketSkip graph, the query is forwarded to the neighbors which are nearest for the range from 600 to 5000. These nodes are checked for query hits and both send a hit (contact A) to the requester, whereas the potential contacts B and C do not fulfill the query. These two PacketSkip nodes are both leaf nodes of the search tree as the full-search range is covered and therefore the requester recognizes that the search procedure has completed. Contact A with its capacity <CPU: 1200 and BW: 4800> is the only peer meeting the query's requirements. For the k-search, the query carries an additional k counter, which is decremented on each hit, and the query is not forwarded any further if the counter reaches zero. Additionally, due to tracer routing, the initiator is able to interrupt the search query, if enough hits were collected. For the hop complexity, it takes $\log(N) + \log(\frac{N}{l}) * \log(N)$ to find the first

relevant PacketSkip node of the dimension within the shortest search range. Then, in worst-case, which is the full range search from $-\infty$ to ∞, we need to traverse all PacketSkip nodes ($=\frac{N}{l}$) in the spanned search tree and we need $\log(N)$ hops to access a single PacketSkip node in the DHT. This results in $\log(N)+\log(\frac{N}{l})*\log(N)+\frac{N}{l}*\log(N)$ hops and is in $\mathcal{O}(\frac{N}{l}*\log(N))$. It must be noted that the average case is much better as not all PacketSkip nodes need to be traversed. Especially, the k-search stops when reaching k hits or the queries' range has been completely traversed without finding k hits, respectively. For the time complexity we expect to finish the search fairly quickly as a lot of paths are traversed in parallel. The search tree, containing $\frac{N}{l}$ nodes in the worst-case, is traversed in a parallel manner needing $\log(\frac{N}{l})$. All taken together, the time complexity is $\log(N) + \log(\frac{N}{l}) * \log(N) + \log(\frac{N}{l}) * \log(N)$ and can be simplified in the worst-case to $\mathcal{O}(\log^2(N))$. Once again, like the complexity calculation for hops, the k-search is expected to finish faster.

In the end, we evaluate our solution and compare it with our previous work CapSearch. For this, we follow the same evaluation methodology like in CapSearch and implemented PacketSkip in PeerfactSim.KOM. We applied the same environment and simulation scenarios and compare the outcome with CapSearch. On the one hand, for the search precision and false positives, we encounter a near optimal behavior for both, k- and full-search, while keeping the time and hops needed low. We observe that the search in PacketSkip outperforms the search in CapSearch in all categories. Furthermore we determine a superior behavior in terms of capacity update in PacketSkip. It performs 3 to 4 times faster than CapSearch and needs 5 to 6 times less messages for this. On the other hand, we observe a worse load distribution in PacketSkip compared to CapSearch, where 11 to 14 % vs. 98 to 99 % of all peers are involved. However, if the number of capacities increases, then the number of DHT objects needed to store the indexing scheme will rise. This factor is smaller for PacketSkip, therefore it scales better than CapSearch for a higher number of capacities. An increasing number of capacities leads to a deeper kd-tree in CapSearch and hence the capacity update process is prolonged, too. In contrast, the duration of PacketSkip's capacity update process is barely affected as capacity announces are split and routed for each capacity space in parallel. Lastly, CapSearch's maintenance traffic is significantly lower than PacketSkip's by a factor of roughly 2 to 4. We state that the traffic is still low in absolute numbers.

5.2.4 Related Work

Previously, in Section 5.1.4, we presented related work in the research field of capacity-based search and the general major area of research for multidimensional indexing and search. In this section we address related work utilizing Skip Graphs, just like PacketSkip does, and our paper CapSearch [34]. The consideration is based on the literature discussed in our paper.

CapSearch, which we already discussed in Section 5.1, builds up a modified kd-tree structure operating atop a DHT for indexing and search peer capacities. We required a static number of capacity spaces and capacity classes to be able to directly access tree nodes. In contrast to the paper CapSearch, here, we use a Skip Graph to map multidimensional onto one-dimensional data. Furthermore, we support a dynamic number of capacity spaces and classes.

Aspnes and Shah laid a foundation for PacketSkip with their paper about Skip Graphs [5]. The authors propose an overlay to let peers arrange data in sequential, thus ordered, lists. To

be able to route efficiently in this overlay, they propose to use 'lists of lists' to create shortcuts to distant nodes. Queries are forwarded to nodes that are closer to the target and per hop the distance is halved, thus, the hops needed is in $\mathcal{O}(\log(N))$. Extending this structure, Xu, Zhou and Guan propose SkipCluster [130], which builds up a super-peer Skip Graph structure to cluster peers. In our work, we use a Skip Graph as a logical node structure stored in the DHT and use homogeneous node roles instead of a hierarchical structure, which does not rely on super-peers. Thus, super-peer promotion mechanisms and the treatment on their potential leaving are not needed.

In the literature several multidimensional indexing methods have been proposed, which use a Skip Graph, and in addition to it a kd-tree to partition the space or space-filling curves to map the data on a one-dimensional space. SkipIndex [133], proposed by Zhang, Krishnamurthy and Wang, counts to the former. The kd-tree's data, which is maintained by its leaf nodes, is stored in the Skip Graph to be able to identify regions to route queries to. Approaches, which belongs to the former category and use space-filling curves, are SCRAP [40] from Ganesan, Yang and Garcia-Molina and ZNet [112] from Shu et al. These use space-filling curves, namely Hilbert curve and Z-curve, to map the multidimensional data onto a one-dimensional space. This data is finally inserted into the Skip Graph and its mechanisms are used to query for data. In contrast, as we are focusing on capacities, which are labeled numbers and therefore live in the space $\mathbb{R}^k$, we map all data onto a one-dimensional space $\mathbb{R}^1$ without using any space-filling curves. By doing so, we do not need to adapt any curve parameters to fit our needs and are therefore more flexible. The Skip Graph is used to maintain the data and its nodes are DHT objects and not physical peers. Our search is simple, lightweight and highly parallelizable and therefore fast.

5.2.5 Personal Contribution

Andreas Disterhöft, the author of this thesis, defined aims for the project, which are the shift from a fixed to a dynamic number of capacities and an improved search accuracy, especially for the full-search. After Andreas Disterhöft did some literature research for current state-of-the-art multidimensional indexing protocols and an overview of such data structures, he and Andreas Funke decided to use Skip Graphs. Andreas Funke did deeper literature research in Skip Graphs and Andreas Disterhöft and Andreas Funke created the conception in discussions, where the major definition of the protocol has taken place. After the specification of the protocol, Andreas Funke implemented it in PeerfactSim.KOM. For the evaluation part, Andreas Disterhöft planned the methodology, the setups and metrics. Andreas Funke executed the simulations on HILBERT, the High Performance Computing System (HPC) from the Heinrich Heine University, using scripts provided by Andreas Disterhöft. The first outcome and its analysis were discussed in meetings. In following discussions, further simulations have been performed for the paper and the analysis was extended by Andreas Disterhöft. Kalman Graffi contributed with discussions about the idea and protocol. The paper was written by Andreas Disterhöft, Andreas Funke and reviews were performed by Kalman Graffi. In particular, Andreas Disterhöft wrote the introduction and the first half of related work, in which we stated related work in the field of capacity-based search and resource discovery. Andreas Funke wrote the second half of related work and the major part of the protocol description (Section III). The evaluation was written by Andreas Disterhöft and the work on the conclusion was again split between him and Andreas Funke.

Chapter 6

Future of Peer-to-Peer Systems? The Web!

In previous sections we assumed that a p2p system is running and focused on decentralized monitoring fields, while, in this section, we elaborate in which environments such p2p systems run currently and where we place them in the future. In the last decades users were invited to participate in p2p systems, if they are willed to install third-party software on their local machine. Thus, the software installation is the first barrier, which hinders new users to try out new systems and participate in its environment. Furthermore, users are deterred by software configurations, which are needed to run the software correctly. This configuration includes the opening of ports in the firewall and its determination in the software. Therefore, p2p software is often understood as software for 'computer geeks'. Moreover, they have a bad reputation because of a lot of file sharing applications were connected with a traditional p2p system. At that time, file sharing applications had copyright issues because participants uploaded copyright-protected content while others downloaded it. Lastly, undecided users may be deterred by potential performance issues compared to traditional client / server architectures. Also, users may fear getting their (upload) bandwidth stolen so they are not able to perform other tasks in parallel.

However, while we were not tackling all fundamental problems in p2p systems mentioned above, we focus on introducing Internet users to p2p systems by simplifying the first contact. Nowadays, Internet users are used to web or mobile applications, whereas the usage of the former does not need any prior installation and the vendors of the latter provide a simple '1-click installation' using their respective app store - Google Play Store, Apple App Store or Microsoft Windows Phone Store. Prominent examples for such applications are social networks (Facebook, Twitter, Instagram), a lot of HTML5 [57]-based web-services (Netflix, mail providers), Skype To Go and online-banking. Netflix ported their video-on-demand streaming service to HTML5, thus, users can watch movies on their devices without installing any further software like flash or alike. Hence, web and mobile services are ubiquitous and are widely used. As users are used to such web services and it needs new innovations to bind users, new standards have been introduced to enable easy-to-use real-time communication using the browser - WebRTC has been born. It started back in 2011 with the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF) which decided to work on WebRTC, whereas the W3C is responsible for the Browser APIs [11] and the IETF for the protocol definition [1]. WebRTC created the possibility to establish direct real-time communication between browser instances. Thus, browser instances are able to communicate to each other without being redirected by a

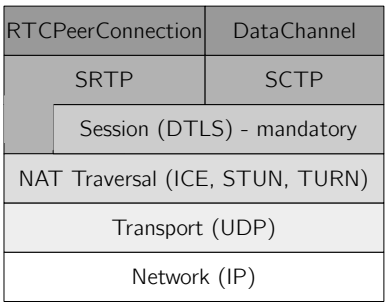


Figure 6.1: WebRTC architecture, see [53] the right-hand side stack of Figure 18-3.

server. The work-flow is quite easy, as a user simply needs a browser which implements WebRTC, visit a website using WebRTC technology and establish communication to other website visitors to communicate in a browser-to-browser manner. This opens the way to peer-to-peer applications running in a browser!

Before introducing our work on this field, we give a short introduction to WebRTC and its techniques. To establish a connection to other (remote) browser instances, one needs to create a `PeerConnection` object and connect to the respective remote object. The WebRTC flow does not dictate a signaling service, thus, the web application developer is responsible for the signaling. A signaling service defines how peers find and exchange meta data with each other, such as the IP address, port and the media meta data. Here, Jingle [76], Asterisk or even a custom signaling service hosted by a server is conceivable. Grigorik [53] summarizes the architecture used in WebRTC in Figure 6.1 which shows the used protocols. The before-mentioned object `PeerConnection` thereby represents a single logic connection between a browser-pair and holds relevant connection objects and meta information. Inter-browser communication is realized atop UDP [94]. Due to NAT issues several NAT traversal mechanisms are used to successfully establish a connection: Interactive Connectivity Establishment (ICE) [102], Session Traversal Utilities for NAT (STUN) [103], Traversal Using Relays around NAT (TURN) [77], RTP-over-TCP [70] and via proxy. After a connection between two browser instances has been established, media (audio/video) or data channels can be opened. To securely deliver data, the session protocol Datagram Transport Layer Security (DTLS) [101] is used, which is mainly the Transport Layer Security protocol (TLS) [31] over UDP. For media transport the Secure Real-time Transport Protocol (SRTP) [9] is used and represented in a `RTCPeerConnection` object. Furthermore, a data channel can be used for arbitrary data and is represented by a `DataChannel` object. Here, the Stream Control Transmission Protocol (SCTP) [114] is used, which provides a configurable reliability and delivery in contrast to UDP and TCP [95]. Making a long story short, signaling service must be implemented by the developer, commonly used NAT traversal protocols are used for hole-punching, DTLS secures data and media channels, whereas the data channel's reliability can be configured and media channels use SRTP which is a commonly used protocol for voice over IP (VoIP).

Another important information must be taken into account to decide whether a new standard is worth working with. The distribution of the WebRTC standard or the number of browsers implementing the WebRTC standard, respectively. The compatibility grid in Figure 6.2 indicates that common browsers, like Google Chrome, Mozilla Firefox, Opera, and Edge, implement

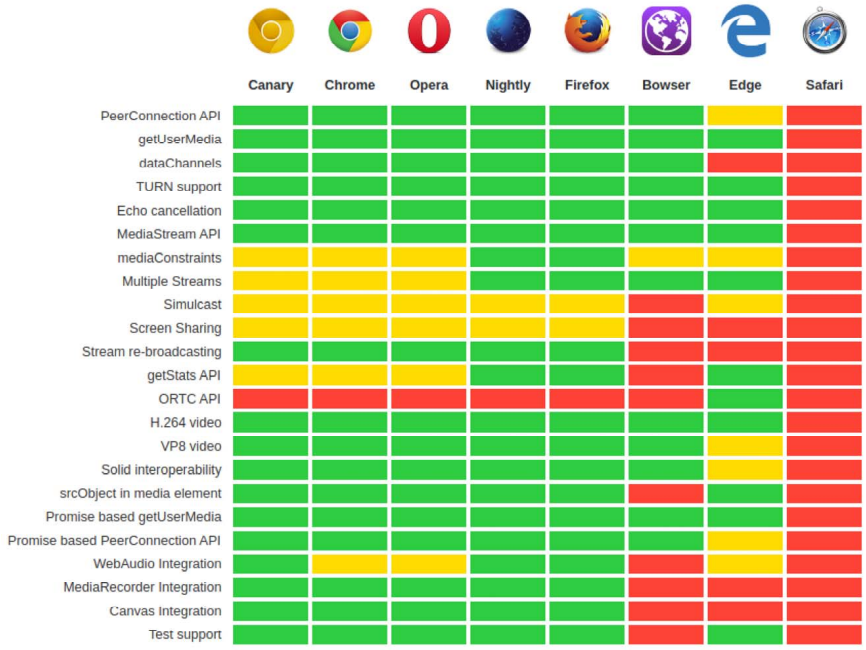


Figure 6.2: WebRTC compatibility grid, see [122].

relevant features from WebRTC. A simplified version of the grid is shown in [123], here[1], one can figure out that almost all common browsers except Safari implement the most important features of the WebRTC standard. Apple already has begun on the implementation in their Safari and Version 11.0 is already supporting it [4], thus, in future all relevant browsers will support WebRTC. This is good news for the future of WebRTC. We conclude that it is worth working with WebRTC and in future it can replace the way we use peer-to-peer software.

In the following we refer to our work in this field, where we aimed to bring p2p systems to the web using WebRTC. Our first work is motivated by p2p-based social networks, so called decentralized online social networks (DOSN), which already exist but have a low number of participants. Datta et al. [30, p.1] define online social networks as "an online platform that (1) provides services for a user to build a public profile and to explicitly declare the connection between his or her profile with those of the other users; (2) enables a user to share information and content with the chosen users or public; and (3) supports the development and usage of social applications with which the user can interact and collaborate with both friends and strangers". We therefore state that the minimum functionality is as follows: First, users should be able to create a profile which identifies them. Second, they should be able to search for other users and store the acquaintances. Third, the service should provide communication channels between those friends or users in general. The motivation on using DOSN instead of centralized online social networks (OSN) is described by works from Buchegger and Datta [16] and Paul, Greschbach, Buchegger and Strufe [90]. The main reasons are privacy and security issues due to the centralized storage of data which goes hand in hand with censorship risks. Known solutions for DOSN are LifeSocial [46], Diaspora [13] and PeerSon [17]. The 'installation

[1] As of 14th January 2018.

barrier' phenomena, which we have mentioned above, could be one of the reasons for these low numbers. With WebP2P [33, 38], which we present in Sections 6.1 and 6.2, we focused on porting a well-known p2p network (Chord) and providing building blocks for a chatting and video streaming platform. By doing so, we aimed to keep the user data safe and private by encrypting it. Furthermore, we are interested whether WebRTC fits the needs of current third-party-based software solutions from DOSNs.

In our second use-case for WebRTC we focus on efficient browser-to-browser (or p2p) video streaming with our solution Chunked-Swarm, which we present in Section 6.3. We propose an efficient live-video streaming software which chunks the live video content and smartly distributes the chunks amongst the participating peers to save costs. The use-case for Chunked-Swarm is as in Twitch.tv, where the authors Zhang and Liu [135] state that, the distribution from viewers to streamers follows a heavy-tailed Weibull distribution, thus, there are a lot of unpopular streams with just a few viewers but a very few streams with a lot of viewers. In Chunked-Swarm we focus on the former group with up to 200 viewers. In our paper we propose this system for p2p environments in general, but this system is predestined to use WebRTC. In fact, Christopher Probst, the first author of the paper, founded the startup StriveCDN [98] with an enhanced version of Chunked-Swarm. StriveCDN stands for an approach operating always at least as efficient as the client / server approach but can also save up to 95 % of the server's upload bandwidth. Whereas, the client side remains simple as it was: The user fires up its browser to watch a live-stream.

In the following, we first present our two papers about WebP2P [33, 38] followed by our paper about Chunked-Swarm [97].

6.1 A Browser-based Secure P2P Framework for Decentralized Online Social Networks

This chapter contains the contribution and a summary of our demo paper WebP2P [33], which is appended as a verbatim copy.

Andreas Disterhöft and Kalman Graffi. "A Browser-based Secure P2P Framework for Decentralized Online Social Networks". In: *Proceedings of the IEEE International Conference on Networked Systems (NetSys)*. 2015.

In the following we state our contribution of the demonstrator WebP2P in Section 6.1.1, followed by its importance and impact on this thesis in Section 6.1.2. A summary is given in Section 6.1.3 and we close this section with the personal contributions in Section 6.1.4.

6.1.1 Contribution

The contribution of this paper is as follows. With WebP2P we present a first demonstrator for a WebRTC-based (video-)chatting platform running atop a p2p overlay. We heavily adapted OpenChord, an open implementation of Chord, to provide social network building blocks, that are the encryption / signing of DHT entries, user search by using self-signed certificates, a deterministic public key / Chord ID creation depending on the username and passphrase and an encrypted communication using ECC. The social platform operating atop provides a login, a buddy list storage, the user search by username or a user's public key and an audio / video and text-based chat.

6.1.2 Importance and Impact on Thesis

This paper's impact on the thesis is quite limited and is therefore less important than the previous papers in the field of decentralized monitoring. However, with this demo paper we showed the importance of the new standard WebRTC for p2p systems. With our proof of concept we motivate the usage of p2p systems in a different environment: the web.

6.1.3 Summary

The main motivation for our paper is to provide a platform for the applicability of browsers as a basis for p2p networks. These p2p networks offer a broad range of functionality and we focus on implementing a social network atop. In our paper we present an easy-to-use secure p2p-based social network in order to overcome the problems of centralized data storage and its misuse. Although solutions already exist, they are only hardly used as they are difficult to use for unexperienced users. Such examples are LifeSocial [46], Diaspora [13] and PeerSon [17].

Although these DOSN offer more functionalities than we do, like a personal wall where (public) posts can be created, we concentrate on a limited set of functionalities to provide a first simple social network in the form of a chatting platform.

We propose our demonstrator called WebP2P, which is divided into two parts containing several modules. Firstly, we build up a traditional p2p network, for example Chord, with browsers using WebRTC. Secondly, we run a social network application atop, which provides the following services. Users can login by entering their username and password, they automatically store a personal buddy list, users can be searched with the help of *self-signed* certificates, the interaction of users is focused on audio-/video- and text-chatting and all data and communication paths are encrypted using public key cryptography (ECC). The underlying p2p overlay, named *WebDHT*, is a heavily adapted version of OpenChord to run WebRTC for the communication between peers or browsers, respectively, and provide a security model for the DHT and the communication on the overlay layer. As proposed by Graffi et al. [49], the Chord ID is set to the public key, which is deterministically generated using the username and passphrase of the user, thus, using the public key cryptography all communications are encrypted and signed. The DHT is used as a secure storage of user-related data. In general, all participants store their encrypted data at well-defined DHT locations. Self-signed certificates are created and stored at username dependent locations in the DHT, thus, other users are able to search for them if the username or its public key is known. These building blocks are used in the (video-)chatting application running atop. For the monitoring of the network, which is only activated for evaluation purposes, we go for a rudimentary centralized monitoring approach. Here, each peer sends its monitoring data directly to a central instance. The GUI module uses the application interface to forward user interactions accordingly or display information obtained by the underlying system, respectively. Therefore, we offer an intuitive and window-based front-end to our users.

6.1.4 Personal Contribution

Kalman Graffi, the second author of the paper, had the initial idea to use the new standard WebRTC to implement p2p overlays with browsers. Andreas Disterhöft, the author of this thesis, and Kalman Graffi had discussions on further steps. Andreas Disterhöft proposed to start with Chord and its implementation OpenChord and adapt it to our needs. Moreover, he defined our needs and proposed mechanisms for the social network: security, resilience, building blocks for the (video-)chatting platform, adapted the Chord join process, user search and the creation of self-signed certificates. The implementation was brought forward by him. Referring to the paper, Andreas Disterhöft wrote it and Kalman Graffi gave feedback and provided reviews.

6.2 Protected Chords in the Web: Secure P2P Framework for Decentralized Online Social Networks

This chapter contains the contribution, a summary and related work of our paper Protected Chords [38], which is appended as a verbatim copy.

Andreas Disterhöft and Kalman Graffi. "Protected Chords in the Web: Secure P2P Framework for Decentralized Online Social Networks". In: *Proceedings of the IEEE International Conference on Peer-to-Peer Computing (P2P)*. 2015. Acceptance Rate: 42.4 %.

The outline of this section is as follows. In the next section, in Section 6.2.1, we state our contribution of the paper. After this, we present the importance and impact on this thesis in Section 6.2.2 and give a summary in Section 6.2.3. We discuss related work in the field of decentralized online social networks in Section 6.2.4 and lastly declare the workload share on this paper in Section 6.2.5.

6.2.1 Contribution

The contribution in this paper is the presentation of our extended version of the WebP2P demonstrator presented in Section 6.1. This extension contains the following three contributions. Firstly, we thoroughly explain the architecture of WebP2P. Secondly, we implemented and simulated user-behavior by applying behavior models for instant-messaging (model from Leskovec et al. [71]) and behavior models for video streaming (from Guha et al. [54]). Lastly, we evaluated WebP2P to rate the feasibility of p2p overlays and decentralized social network applications running in the browser utilizing WebRTC.

6.2.2 Importance and Impact on Thesis

In prior work we presented a demonstrator of WebP2P showing a proof of concept, whereas we evaluated its performance in this paper. This is an important basis for other p2p-based applications whose developer plan to port their standalone software. In the light of the foregoing, we contribute to the future of p2p systems in regard of their feasibility in the web environment. This may have an impact in the future, at least we verify that the complexity of installing and configuring p2p software is successfully omitted.

6.2.3 Summary

In this paper we extend our demonstrator of WebP2P by in-depth descriptions and an evaluation of its performance and costs.

The detailed descriptions of our architecture and its components are as follows. We give an overview which parts in OpenChord we adapted and which parts we added. For this, we refer to Figure 2 in the paper. The join procedure needed some adaptation and got a new bootstrap method to enter an existing Chord network or create a new network. Furthermore, we detail the introduced DHT entries, namely, unsigned, signed and signed_encrypted entries to provide a basis for the private and public application storage. Signed data at well-defined positions in the DHT is used to provide an user search either by public key or by username.

Besides the desktop GUI, we implemented a mobile GUI for participating smart phones and we implemented a user behavior simulation by implementing the GUI interface. For the user behavior we applied models proposed by Leskovec et al. [71] and Guha et al. [54]. The former work models the instant-messaging behavior and the latter the behavior in video streaming sessions. This user simulation is used as workload in the evaluation where we evaluated whether the requirements of WebP2P are feasible. For this purpose 50 simulated participants were evaluated in the lab. We conclude the outcome as follows. WebP2P, which includes a p2p network and a social network running atop, is feasible in terms of bandwidth consumption and CPU load but a few tweaks can be applied to boost the platform's performance even further and lower the costs of the application.

6.2.4 Related Work in the Field of Decentralized Online Social Networks

In this section we examine related work which is based on the paper's related work section. Our solution WebP2P can be classified as a decentralized online chatting platform, which is a subset of decentralized social networks, as the latter contains amongst others a user specific wall with (partly) public posts, group memberships etc. The group of decentralized social networks can be divided into private server approaches and p2p-based approaches.

In the first group, the private server approaches, nodes in the overlay network are servers that are always online and users connect to servers to enter this network. Examples are Diaspora [13], FOAF [43] and Persona [7]. In the second group, the p2p-based approaches, nodes in the overlay network are participants who are subject to churn. This means that they go on- and offline at their own will. In the literature this group is further subdivided into friendship-based and DHT-based overlay structure. Solutions for the former category have the benefit that routing requests can often be resolved locally and a certain trust model is given by storing data on friends. Examples for this category are Safebook [24], GoDisco [29] and the decentralized social networks overlay proposed by Mega, Montresor and Picco [82]. Problems arise when it comes to bootstrapping participants with few friends. If these friends are offline, a foreign user has to be chosen. This implies another problem, the risk of data availability. If all friends of a certain participant and the participant itself live spatially close, the data will not be reachable once all friends go to sleep at night. Therefore the data needs to be stored at another position, for example, on a foreign participant. Another problem arises concerning the trust level between friends. For this, one has to ask himself whether he trusts all of his friends in social networks, such as Facebook. Most likely this question has to be answered in the negative.

The other category, the DHT-based overlay structure, is based on a DHT storage and routing without caring about friend network structures. This eliminates the data availability problem

as the data can be stored anywhere in the DHT as access control is applied, therefore, unauthorized users are not able to read personal data. Examples for this category are LifeSocial [46], PeerSoN [17], SuperNova [110] and OverSoc [125]. Although an access control is applied for group access to (personal) files, these models are quite complex and add an overhead.

Our solution WebP2P is located in the category of DHT-based p2p approaches and uses encryption to store private data. Access control is not needed as data located in the DHT is not shared between participants. Prior to overriding DHT objects, participants check whether the requester is the owner of the existing object. Based on this, it grants or denies the permission to override the object.

6.2.5 Personal Contribution

Due to the fact that this paper is an extension of the WebP2P demo paper, the workload share of the personal contribution from Section 6.1.4 applies here, too. In addition to that, Andreas Disterhöft, the author of this thesis, planned, implemented required modules and executed the evaluation in the lab. In particular, he adapted the monitoring module and implemented the simulated user behavior as a GUI module. The paper was written by him, whereas Kalman Graffi gave feedback and provided reviews.

6.3 Chunked-Swarm: Divide and Conquer for Real-time Bounds in Video Streaming

This chapter contains the contribution, a summary and related work of our paper Chunked-Swarm [97], which is appended as a verbatim copy.

Christopher Probst, Andreas Disterhöft and Kalman Graffi. "Chunked-Swarm: Divide and Conquer for Real-time Bounds in Video Streaming". In: *Proceedings of the 15th IEEE International Conference on Next Generation Wired/Wireless Advanced Networks and Systems (New2An)*. 2015. Acceptance Rate: 35 %.

This section is structured as follows. First, we state the contribution of our paper in Section 6.3.1 and present then its importance in the literature and impact on this thesis in Section 6.3.2. We give a summary in Section 6.3.3 and discuss related work in the field of live video streaming in Section 6.3.4. We close with the declaration of the workload share in Section 6.3.5.

6.3.1 Contribution

Our contribution in the paper is divided intro three parts. Firstly, we propose Chunked-Swarm, a protocol to distribute content amongst interested parties in a swarm-based manner. Its attributes are as follows. We introduce a smart chunking distribution model and give distribution guarantees, which is novel, as prior work focused on best-effort approaches. Secondly, we implement Chunked-Swarm as a Java application. Thirdly and lastly, we evaluate the implementation and show that the guarantee can be held in practice with few hundred participants and the costs are significantly reduced. Relevant use-cases, which profit from distribution guarantees, are live-streaming scenarios.

6.3.2 Importance and Impact on Thesis

At first glance, this paper seems not to fit into the topic of enhancing the applicability of p2p applications and the influence new web standards have on it. However, after the publication of this paper, work on it has continued; a port to WebRTC has taken place and several other adaptations have been made. Now, this approach is the main software of a startup called StriveCDN [98]. So, a closer look reveals that the p2p streaming approach and WebRTC are eligible for real world applications. Therefore, we again support our statement from previous sections: WebRTC helps to bring p2p networks to the users and the industry.

6.3.3 Summary

In this paper we propose Chunked-Swarm, a p2p live-streaming protocol to lower the load of the content-provider by utilizing the watcher's upload bandwidth. Furthermore, we guarantee minimum streaming delays in contrast to state-of-the-art live-streaming approaches which offer a best-effort service.

Chunked-Swarm works best in scenarios with a few hundreds viewers. The popular video streaming platform Twitch.tv, where mainly gamers stream and viewers are invited to join their channels, is a very good match and use-case for Chunked-Swarm. In Twitch.tv, according to the measurement study performed by Zhang and Liu [135], around 85 % of all streamers have less than 200 viewers. So, in the first place we focused on this group to provide an efficient live-video streaming approach to lower the bandwidth consumption on the server-side.

The main idea applied in Chunked-Swarm is visualized in Figure 6.3 and is as follows. In a traditional client / server scenario, the streamer sends its stream to the server and the server distributes it among its attending viewers. We adapt this by adding a further step and therefore increasing the streaming delay by a calculable constant but decreasing the bandwidth consumption on the server side. This is achieved by chunking the data and sending one unique chunk to each client and these clients then push their unique chunk to the other clients. Other streaming services are conceivable where the streamer does not need to have a high upload bandwidth as it needs to upload the content only once regardless of the number of peers.

We now detail the distribution plan based on Figure 6.4. First, we assume, that the streamer's and viewers' download rate is higher than their upload rate and all participants (streamer and viewers) have the same down- and upload bandwidth. Each viewer joins the swarm by contacting the streamer, which has a full list of all viewers in this swarm. Figure 6.4(a) reveals the first step: the streamer or seeder, respectively, chunks its first content part in chunks and sends one disjunct chunk to each peer. This takes the same time it needs to upload all chunks to only one peer. By doing so, the whole first content part is distributed amongst the peers. Step 1 is finished, which takes T_0 seconds. The second step is shown in Figure 6.4(b). The seeder continues with the next content part (two), chunks it and sends it to its peers as shown in Figure 6.4(a). At the same time, all peers upload their disjunct content part (one) to every other peer in this swarm. Due to the assumption that up- and download bandwidth of seeder and peers are the same, this takes another T_0 at maximum. Hence it takes $2*T_0$ to distribute this part to all peers regardless of the number of peers. Furthermore, the maintenance of the overlay increases with the number of involved peers as they are connected in a mesh structure. Lastly, Figure 6.4(c) shows the peculiarity of the live streaming. It is a special case as the distribution is limited and focuses on the last $2*T_0$ as all other parts are too late and can therefore be discarded. All in all, in Chunked-Swarm a tree with a fixed depth of two is created. Other approaches use a deeper tree and therefore can not give any delay guarantees.

In the evaluation of our approach, we implemented the protocol as a standalone software in Java and provided a proof of work in several scenarios. Furthermore, we analyzed whether the $2*T_0$ delay boundary can be fulfilled in practice and whether two consecutive content parts interfere. In the outcome we summarize that we are able to fulfill the $2*T_0$ delay boundary in practice, whereas T_0 is the time needed to transfer one un-chunked part to the slowest peer. Two parts do not interfere, which is a very good characteristic for live-video streaming. Lastly,

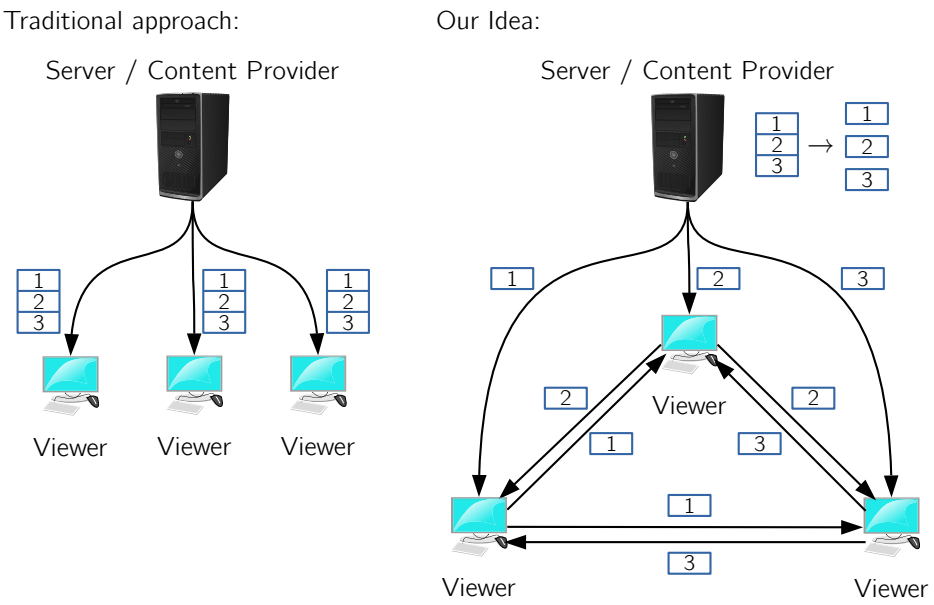


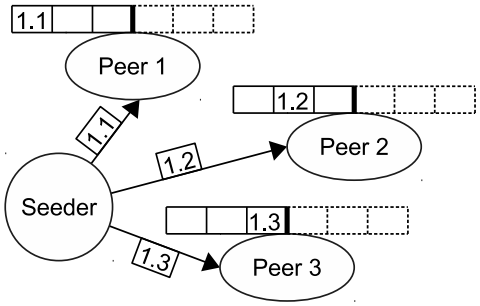
Figure 6.3: Comparison between a traditional approach and our main idea. While in the former approach the server needs to send the whole part to each of its clients (viewers), in our approach the server chunks the whole part and it needs to upload it only once. The viewers further distribute the missing chunks between themselves.

the maintenance overhead depends on the number of peers and is reasonable for a few hundreds spectators due to its mesh structure. To sum up, Chunked-Swarm is an approach, which utilizes unused upload capacities of the participants, and is therefore advisable for 85 % of Twitch.tv's streamers, other streaming services and use-cases requiring guaranteed distribution plans in general to lower the operational costs of the company.

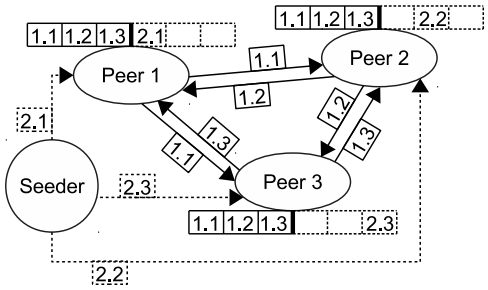
6.3.4 Related Work in the Field of live Video Streaming

In the following we refer to the research field of live-video streaming. This section is based on the paper's related work section. Our main focus lies on approaches in p2p networks, which allow to stream to a large number of participants providing low costs for content delivery but introducing a certain amount of complexity. Before heading to the *full-distributed* scheme, we discuss *centralized* and *clustering recursive* schemes in the following.

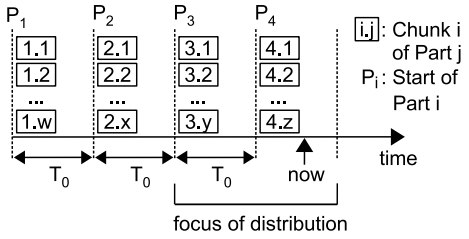
In *centralized* schemes a central instance exists for the coordination of the communication between peers. These peers are typically organized in a tree having a height of $\mathcal{O}(\log(N))$. Having a central coordination unit increases the efficiency of the communication and reduces delays but the central unit remains a single point of failure and is subject to high load in large structures. Examples for such approaches are ALMI [91] and CoopNet [87]. The authors Graffi, Kaune, Pussep, Kovacevic and Steinmetz [47] focused in their work on a DHT-based approach to utilize the heterogeneity of peers to match them for ideal transmissions, thus, by taking their characteristics into account.



(a) First round: Seeder sends disjunct chunks of the first part to all peers.



(b) Second round: Seeder sends distributes the second part while the peers distribute the missing chunks of the first part among themselves.



(c) Time model of the live-video streaming.

Figure 6.4: Distribution models in Chunked-Swarm taken from our paper [97].

Clustering recursive schemes in turn build up trees of clusters or swarms and therefore provide a multilayer structure. However, these schemes are similar to regular trees in terms of the structure but they increase the complexity of the system. Although the delay is decreasing, the height of the tree remains $\mathcal{O}(\log(N))$. Popular representatives are NICE [8] and THAG [117].

Lastly, *full-distributed* schemes are subdivided into swarm- and tree-based approaches. On the one hand, swarm-based approaches chunk their content, thus, the content is divided into small pieces and all peers try to obtain all chunks. No particular structure, like a tree, is applied. Instead, peers hold bitmaps of obtained chunks and exchange these with other peers to find and eventually initiate a download to retrieve missing parts. Famous solutions are BitTorrent Live [106], Chainsaw [88] and LayerP2P [74]. On the other hand, in tree-based approaches the root node is the content provider or seeder, respectively, and the other inner nodes and leaves are streamers. The downside of these approaches is the fact that the leaf nodes' upload capacities are not utilized as content is distributed top-down. The positioning of the nodes in the tree is either supported by the DHT, network-driven or data-driven. Examples for the DHT-supported positioning are SplitStream [19] and Bayeux [138]. However, the DHT support induces a maintenance overhead. The network-driven approach builds up a fat-tree to optimize the data flow, thus weaker peers do not get overloaded. So, peers with high bandwidth capabilities are located at higher levels in the tree. Network condition dynamics are supported, too. Network-driven approaches are mTreebone [121] and TreeClimber [136]. In contrast, data-driven trees are organized according to the position in the video playback. The root node is the one who is ahead of all as it is the seeder, peers in the tree's level 1 are

just behind the root but ahead of their children in level 2, and so on. Prominent data-driven solutions are CoolStreaming [129], Substream Trading [75] and SPANC [21].

However, our solution is a hybrid between swarm- and tree-based full-distributed approaches. We propose a novel attribute: the video delivery deadline guarantee.

6.3.5 Personal Contribution

Andreas Disterhöft, the author of this thesis, contributed in discussions about protocol details, the use case, the evaluation methodology and its analysis. In these discussions the other two authors, Christopher Probst and Kalman Graffi were involved. Christopher Probst had the initial idea, proposed the first protocol design draft and implemented the prototype. The workload share for the paper is as follows. Andreas Disterhöft wrote parts for the introduction, related work and evaluation and provided reviews and feedback. Kalman Graffi worked on the related work section and provided reviews and feedback. Christopher Probst, the major contributor of this paper, wrote the section about Chunked-Swarm and large parts of the other sections.

Chapter 7

Conclusion and Future Work

In this thesis we focused our attention on p2p networks in general and their monitoring solutions in particular. For this, we defined five problem statements (PS1 – PS5) together with research questions that have been answered in this dissertation. We motivated the significance of monitoring systems in state-of-the-art p2p networks, which is, the importance of being able to gather the current state of the network to be able to react to deviations to a certain quality of service. Furthermore, by monitoring decentralized networks and evaluating its outcome one can improve its performance, reduce the arising costs or even lay foundations to fix bugs. Our concerns are complex and we contributed on several levels in current monitoring solutions. Firstly, we concentrated on improving existing system-specific monitoring solutions to bring them closer to optimal attributes, which are a high robustness, high accuracy and high efficiency. However, the improvements are and will remain trade-offs between the performance and its costs but moving solutions' key data in terms of performance towards realistic scenarios is our main desire, whereby the costs should remain manageable. Further on, in our elaboration we dealt with two restrictions, which are the problem of partial participation and the monitoring's security. For the former, we motivated the problem of partial participation in a sense that monitoring solutions lack dealing with peers participating in the p2p network but not in the monitoring solution. For the latter, we stressed the importance of tailored security mechanisms to mitigate the influence of maliciously behaving peers which try to free-ride, manipulate or even disrupt the monitoring's mechanisms. For both restrictions we presented and evaluated solutions to overcome the restraint. Beside the system-specific monitoring we devoted to peer-specific monitoring, which is the indexing and search for peers' capacities. We motivated and contributed to this field with solutions that depend on the actual use-case. Lastly, we turned our attention to browser-based p2p networks, which are conveniently and seamlessly usable, especially to the question how to access such networks. In this area, we first identified a problem in the way these p2p applications are accessed and its consequent barriers for new users. We propose to make use of WebRTC for new p2p applications and a port from a standalone application might be meaningful, too.

In the following we detail our conclusion in the above mentioned fields by referring to each problem statement and briefly describing our solutions and their most meaningful outcome. After this, we discuss future work in these fields and close this thesis with a few closing remarks.

7.1 Conclusion

In this section we conclude the work of our papers in the field of system-specific, peer-specific and restrictions in monitoring solutions. Further, we elaborate a convenient and simple form to deploy p2p networks. Now, we turn our attention to one problem statement after the other and conclude our work done in this area.

PS1: Increasing Robustness of structured Monitoring approaches collecting system-specific Data For our first problem statement we proposed an extended version of **SkyEye.KOM** [45], which was initially presented and discussed in the dissertation from Graffi [52]. SkyEye.KOM is a tree-based p2p monitoring system which monitors system-specific information and operates on a layer atop a DHT overlay. The DHT is used for the schedule of responsibilities in the monitoring tree. Monitoring data is gathered in a bottom-up fashion, whereas the data is aggregated on each level, and the dissemination of the global view is performed in a top-down fashion. This initial solution lacks robustness in highly dynamic networks. As a result, when peers dynamically join and leave the network, the tree topology must be reconstructed, which leads to eclipsed sub-trees and eventually to data loss. Our extension increases the robustness of the tree structure by introducing additional communication links. These links are the interconnection between siblings on tree levels near the root node and a volatile connection between the new and old node for a certain position in the tree. The former is used to quickly obtain first information from the siblings as we assume that the monitoring tree is balanced and we therefore assume that the data is similar. While the latter is used to provide the newly arrived node with first information of its position by handing over data gathered by the previously responsible node for this position.

In the evaluation we conducted a parameter study and compared the performance and costs of SkyEye.KOM and the gossip-based Symmetric Push-Sum Protocol (SPSP) [14]. Without churn we verify a superior precision, adaption rate and low monitoring costs for SkyEye.KOM. Under KAD churn [113], the adaption rate worsens and we observe a few overestimations but we can still certify a high precision having an error between 1 – 4 %. The SPSP's adaption rate and precision is not significantly affected. An increase of the monitoring bandwidth consumption is perceived for both, SkyEye.KOM and SPSP. While we recognize an increase of up to 100 % for SkyEye.KOM, we see an increase of around 33 % for SPSP. Still, SkyEye.KOM consumes only around 30 % of SPSP's bandwidth. For the overall network layer bandwidth, we notice that SkyEye.KOM increases its bandwidth consumption by 12 – 26 %, whereas SPSP perceives an increase of 4 % in average. In sum, SkyEye.KOM consumes 36 – 40 % more bandwidth than SPSP, which is due to a missing caching mechanism for SkyEye.KOM and the choice of a low update interval. For each update interval one DHT lookup has to be performed, especially in low churn scenarios we could save a lot of bandwidth when caching neighbors. SPSP on the other hand rarely uses DHT lookups as it has a smart neighbor propagation mechanism. However, when doubling the update interval for SkyEye.KOM, the costs are halved while keeping the precision and adaption rate at a high level and still outperforming SPSP.

While the extended version of SkyEye.KOM provides a high accuracy in moderate churn scenarios, for example when facing churn behavior like in file-sharing applications based on KAD [113], we perceived a severe deterioration when facing stronger churn. We infer that this is due to the starvation of single nodes near the root node in the tree and probably due to the probabilistic approach to consult the siblings' data for further monitoring. To circumvent this weakness we propose the following solution.

With our contribution **Multiple Realities Tree (Mr.Tree)** [39] we enhance the robustness of a class of existing tree-based monitoring solutions by introducing redundancy in the tree. We propose to utilize parallel trees providing pairwise distinct peers for the positions in the tree. Our approach is similar to the approach MultipleTree [10], which is a reactive approach and therefore does not need a mechanism to maintain a shared transient data storage for peers on the same positions in different trees. We propose such a mechanism and peers on same positions in different trees are logically connected forming hypernodes and store and share a transient monitoring data set, which is the main source of robustness. We propose different modes for Mr.Tree:

- *Fixed*: We provide a fixed number for the built up parallel trees.
- *Adaptive*: We provide a minimum and maximum number of built up parallel trees. These trees are dynamic as they are pruned and built up according to the level of churn.
- *Slotted Adaptive*: Additional to the adaptive mode, not all parallel trees send their monitoring data up the tree but in a well-defined order and only if the preceding tree has not successfully delivered its data.

Finally, we evaluated Mr.Tree by applying it atop SkyEye.KOM and comparing it with vanilla SkyEye.KOM and the extended SkyEye.KOM version described above. We defined that the precision is the completeness of data gathered in the global view. Therefore, by linking the precision with the number of lost data sets in the global view, we are able to compare the accuracy of two approaches without bothering about the metric being monitored. We elaborate that a higher robustness for tree-based monitoring systems means that it can achieve higher precision in more dynamic environments.

We first conclude that certain modes of Mr.Tree perform always better than extended SkyEye.KOM in terms of precision and cost-benefit efficiency. For our proposed modes we conclude the following. The slotted adaptive mode saves bandwidth and is therefore the best choice for cost-efficiency. It achieves a 1.3 cost-benefit coefficient achieving 4.6 fold higher precision than vanilla SkyEye.KOM by consuming the 6 fold costs for low churn scenarios. For high churn scenarios the cost-benefit coefficient improves to 0.59 and is therefore cost-efficient achieving 3.2 fold higher precision than vanilla SkyEye.KOM by roughly doubling its costs. In absolute numbers this means that we only induce a relative error of 7 % in contrast to a relative error of 23 %. The adaptive mode is the best trade-off between accuracy and costs in scenarios with higher churn. Therefore we achieve a cost-benefit coefficient of only 1.95 in low churn scenarios (6.1 fold higher precision and 11.9 fold costs) while improving to 0.43 in high churn scenarios (6.3 fold higher precision and 2.7 fold costs). Moreover, the fixed mode is suitable for scenarios facing very high churn and when deviations are not tolerated. The precision gain is ranging from 6.8 fold in high churn scenarios to 5.6 fold in low churn scenarios. Due to the high traffic, ranging from the 25.3 fold to the 4 fold of costs, this mode is only cost-benefit efficient in higher churn scenarios. Lastly, we state that, even when the dynamics in the network are that high, that the DHT overlay perceives a high lookup failure rate, Mr.Tree is able to achieve cost-efficiency.

PS2 / PS3: Monitoring with Restrictions In the second and third problem statement we referred to restrictions or difficulties, respectively, in monitoring solutions, which are concluded in the following.

Influence of Malicious Behavior - The restrictions in monitoring mechanisms caused by maliciously behaving peers, namely Problem Statement PS2, is tackled by our paper **Convex Hull Watchdog** [35]. The problem statement deals with the presence, the influence and the impact of maliciously behaving peers in tree-based p2p monitoring solutions. The influence on the solution is disastrous as they can manipulate the global view or even disrupt the whole system. Thus, attackers can induce errors which sum up to the 1000 fold of the original value, rendering the monitoring data useless. In our paper, we defined an attacker model where peers aim on the manipulation of data, apply disruptive behavior or just free ride the system. We proposed a passively working solution called DOMiNo, which is a service running atop tree-based monitoring solutions. These solutions require a deterministic tree generation and monitoring data needs to be pushed bottom-up. DOMiNo provides two main mechanisms to mitigate malicious behavior. Firstly, we propose to mitigate manipulation attacks by letting inner peers of the tree apply modified Z-Scores and therefore apply an outlier detection. We define it as a convex hull, which is based on the data gathered from other tree branches and therefore attacks should be limited to this convex hull. If a certain incoming data set originated by a source is outside of this convex hull for several times, we propose to drop the data to not let it negatively influence the global view. Secondly, attacks to the tree structure are mitigated by validating the incoming data set's source. If the source's ID does not fit the defined bounds, we propose to drop the data immediately.

For the evaluation we applied our approach atop SkyEye.KOM, let a share of all peers make use of the attacker model and concentrated on the monitoring accuracy, the number of discarded data sets and keep track on the influence of evil behaving peers on the global view. We conclude that we are able to prevent the counting to infinity problem caused by malicious peers introducing loops in the tree. These loops are successfully broken up by our structural mitigation mechanisms. Furthermore, we are able to shrink the relative error introduced by manipulated data sets from 11 – 998 % to only 2 – 18 %, which is remarkable. Hence, we are able to mitigate the influence of manipulation attacks by forcing manipulated values to fit in the convex hull.

Partial Participation - In the second restriction in tree-based monitoring solutions we tackled the problem of partial participation in such systems, namely Problem Statement PS3. To recap, state-of-the-art tree-based p2p monitoring solutions assume that all peers participating in the DHT overlay, also participate in the monitoring system. This is not necessarily true, as participants in such systems are responsible to update their software and therefore certain features, like the monitoring, can be released in a new software version. The implications on tree-based monitoring system are fatal and ultimately lead to data loss due to peers dropping monitoring messages they can not handle. We propose **Minicamp** [37, 36], a solution building up a middleware overlay. This middleware can be run between a KBR-compliant p2p overlay with a few additional requirements and a monitoring overlay. The heart of this service, the middleware overlay, contains only active peers which are responsible for passive peers falling in their responsibility area. This responsibility area is mapped on the p2p overlay and therefore passive nodes are meant to be objects in the middleware. We provide algorithms for active nodes to keep their responsibility range up-to-date by scanning and maintaining it. Concur-

rently, active nodes probe their assigned passive nodes to obtain monitoring information, such as their available bandwidth. This information is gathered in a local data base and used as an input for the monitoring service running atop our middleware Minicamp.

For the evaluation we implemented Minicamp, ran SkyEye.KOM atop and different p2p overlays underneath. We summarize the outcomes as follows. The minimum correctness of the scanning and maintaining mechanisms is 90 % under KAD churn, whereas the false positives reach 5 % at maximum. We state that the monitoring precision and adaption rate almost solely depends on the probing mechanisms. Furthermore, a nice side effect is that a highly accurate probing mechanism not only increases the monitoring's accuracy but also the adaption rate, which can be even higher than having no passive peers. This is due to the fact that less peers, more precise, only the active peers, participate in the monitoring tree, which leads to a lower height of the tree. On the other hand, the per peer load is increased as these peers must maintain the middleware and probe passive peers. Lastly, we conclude the costs for maintaining the middleware. We define a stable state which is reached if an active peer only needs to initiate six lookups per minute. This limit is reached by 10 % active peer share scenarios and above. For the scenario with only 1 % active peers, we perceive up to 20 lookups per minute, which might overload weak peers and mechanisms are needed to overcome the high load.

PS4: Monitoring Peer Capacities: Capacity-based Search In this paragraph we conclude our work done on the field of monitoring peer-specific information. We defined monitoring as a service which gathers certain information and makes the evaluated view accessible to the participants in a communication network. Hence, in this part we concerned about peer-specific information, that is, the peers' capacities. This information is gathered and made available in a structure, which is mentioned as the heart of these solutions. Peers can query this structure to obtain data, therefore the evaluated view can be accessed by a pull-based mechanism. The main challenges a potential solution must take into account are due to the nature of peers' capacities, which are highly dynamic and not aggregatable. In the following, we summarize our two proposed solutions and give the evaluation of them at the end.

Our first solution is **CapSearch** [34], a service which makes use of a modified kd-tree for the indexing structure. This structure is stored in the DHT at well-defined and globally known locations. To prevent data loss, we propose to run a DHT replication mechanism in parallel. The number of capacity spaces and capacity classes per capacity space is fixed and therefore fulfills the requirements of Variant 1. These information are used to initially build up the whole kd-tree, so that, the leaf nodes of the tree represent single combinations of the capacity classes. This means that peers can access them by a single DHT lookup. To insert the capacities into the structure, peers periodically announce their current capacity to the corresponding leaf node. Inner tree nodes of the kd-tree hold information about leaf nodes holding at least one peer. Hence, the capacity search is equivalent to obtain the view of inner tree nodes and traverse the obtained list. For the complexity of the search we estimate a worst case time complexity of $\mathcal{O}(\log k * \log N)$ for the search containing k contacts and N being the number of peers in the network. It comprises one DHT lookup for the inner node and the traversal of a temporal search tree containing k contacts. The capacity announcement takes in worst case the whole tree's depth, which is $\sum_{i=1}^{n} \lceil \log_2 k_i \rceil$ with n being the number of capacity spaces and k_i being the number of capacity classes of the corresponding capacity space.

To overcome the limitation of CapSearch having the number of capacity spaces and capacity classes fixed, we introduced a second approach called **PacketSkip** [32] which fulfills requirements of Variant 2. This solution uses a Skip Graph for data maintenance which is, analogous to CapSearch, stored in the DHT. As already mentioned, the number of capacity spaces and the number of capacity classes are unbounded, because we map the multidimensional capacities to a one-dimensional space by storing their value and a label representing their dimension. For example, <1500 , CPU> represents a 1500 MHz CPU and <500 , UBW> stands for 500 bit/s upload bandwidth. The units are known due to the context. Peers responsible for such a PacketSkip node are responsible for the maintenance of this particular Skip Graph node. For the maintenance, split and merge methods are proposed to increase or decrease the number of PacketSkip nodes when a certain threshold is reached or undershot, respectively. When running this service, peers in the p2p network periodically announce their capacities by bootstrapping the PacketSkip Skip Graph, that is, they first obtain an entry node, and then initiate a search operation to find the right place in the Graph to store their data. The search for capacities is similar to the announcement of data. Hence, the worst-case for the search operation and the capacity announcement can be estimated with $\mathcal{O}(\log^2 N)$.

We evaluated both approaches in our papers and compared both solutions. In churn scenarios CapSearch provides a high accuracy k-search with 90 % precision at minimum and for the full-search the lower limit of 58 % precision is not undercut while keeping the false positive rate always under 1 %. PacketSkip, on the other hand, provides a superior search accuracy and does not fall below 98 % precision and its false positive rate is capped with 2 %. Furthermore, PacketSkip needs significant less hops per search operation and, especially for the full-search, the duration is shorter. For the capacity announcement PacketSkip needs less hops (5 – 6 fold) and offers a shorter duration (3 – 4 fold) but it also induces significantly more maintenance traffic (2 – 4 fold). Furthermore, the load distribution in PacketSkip is worse than CapSearch's with 11 – 14 % vs. 98 – 99 %. In summary, CapSearch is a good choice in scenarios where the fixed capacity space is not a disqualifier, users mainly use the k-search, the smaller bandwidth utilization and the fair workload distribution are important criteria. Otherwise, PacketSkip is the better choice.

PS5: Future of Peer-to-Peer Systems? The Web! In the last problem statement, we bothered about the inconvenience of accessing current p2p applications, that is, the installation and configuration of such third-party software. Potential new users may perceive this as a barrier to test new software. This barrier can fall when entering known waters. Therefore, we analyzed the applicability of browser-based p2p applications. In the following we conclude our work on this problem statement and refer to our papers about WebP2P and Chunked-Swarm.

Our first paper in this field is **WebP2P** [33, 38], a first proof of work to let p2p networks run in a web environment using WebRTC. We ported and heavily modified OpenChord to use the browser environment and WebRTC as its communication protocol. Additionally, we added security mechanisms, a certificate infrastructure together with the opportunity to search for users and the ability to send and receive application-layer messages. We describe these modules as building blocks for social networks. By using these building blocks we implemented a social platform application atop which provides users a login via username and password, few maintenance mechanisms for users' buddy lists, environment for chatting and video streaming.

To evaluate WebP2P we conducted a scenario with 50 peers running on 13 machines and these peers simulate real social media users by applying models taken from the literature [71, 54]. For the load share we observe that streaming takes the largest share with 78 %. Another 11 % are from application-related storage and the Chord-related maintenance takes only 8 %. In particular, the averaged and aggregated bandwidth is as low as 44 KBit/s, which is easily covered by the majority of current Internet users' accesses. Users can perceive a fluent GUI interaction, as the response time for relevant operations takes less than one second. Hence, we conclude that the p2p network using only browsers works fine thanks to WebRTC. The consumed resources are low, technical restrictions exist but are easy to deal with.

Our second paper for this problem statement is **Chunked-Swarm** [97]. Nowadays, the successor of Chunked-Swarm also uses WebRTC and is thus a living use case. However, with our paper about Chunked-Swarm we propose a smart p2p distribution algorithm which guarantees to deliver content amongst its participants in under $2 * T_0$, whereas T_0 is the time needed to push the data to a single participant. The distribution plan is as follows: First, the content gets chunked into the number of participants chunks. Then, the seeder or content maintainer, respectively, sends disjunct chunks to the participants. These participants then distribute their chunk to the others. Finally, after this step, all participants received the content. We emerged an interesting use-case for this distribution algorithm, which is the live streaming of (video) content. By doing so, the data does not need to be held too long in the network and therefore outdated content can be dropped.

For the evaluation of Chunked-Swarm we implemented it in a standalone Java application and provided a proof of work. We conclude that the guarantee of $2 * T_0$ can be held and two consecutive parts of the content do not interfere during the distribution of them, which is a very good feature for live video streaming. Scenarios showed it works best for a few hundreds of peers. However, nowadays this software has been ported to WebRTC and works awesome in a startup, which concentrates to reduce the bandwidth consumption during live video streams.

7.2 Future Work

After having concluded our work and this thesis, we refer to future work on this area of research. Please note, that some of the suggestions are from our papers and are described into more detail in the papers and others are new and came up while writing this thesis.

Firstly, lets turn the focus on system-specific monitoring systems. As mentioned in Chapter 1, our introduction, we went the path to enhance the robustness of current tree-based systems as we saw the greatest potential in them. In the future other structures can be investigated, namely, enhancing the precision of gossip-based or enhance both, the precision and robustness, in hybrid approaches. Going the former way, it needs efficient mechanisms to adaptively decide when to start a new epoch to avoid the span of time the approximation reached the global value and keeps this value for the rest of the epoch. Another reasonable research direction is to introduce a more efficient strategy or topology, respectively, which is used to communicate with a specifically selected neighbor instead of a randomly selected neighbor. By doing so, a communication structure can be built up and redundant communication can be reduced or even eliminated. Coming back to our contribution Mr.Tree, the influence of malicious behavior

should be analyzed as few new attack vectors were opened through the introduction of logical nodes. We should start with our contribution Convex Hull Watchdog and examine the influence of a much greater data base on the efficiency of the solution, which is expected to be of a positive nature.

Besides the monitoring of system-specific data, it can be helpful to create a monitoring on the documents located in the DHT. This monitoring approach can build up rankings of decentrally occurring events. These events might be the the number of accesses, its consumed bandwidth and their rate of change. Being able to access the k^{th} entry of the ranking, one can improve the p2p experience by taking this data into account to predict bottlenecks and support peers that host a very famous document or use the data to analyze the next trend and therefore serve similar documents for example.

Further future work can be done in the field of security in monitoring solutions. Firstly, we could loosen the passive requirement for our Convex Hull Watchdog and propose a mechanism that reacts on manipulated data with a reconstruction process. By doing so, an identified malicious peer can be circumvented by involving its children or its parent, respectively. Furthermore, mechanisms from the machine learning literature can be taken to predict future monitoring results. First study showed that the training of the mechanism turns out to be difficult due to the missing desired state. Thus, further investigation on this topic could be interesting.

Regarding the partial participation of peers in monitoring solutions and our solution Minicamp, we advice to further optimize the approach. On the one hand, costs active peers have should be reduced, especially when only a small share of all peers are active. On the other hand, the responsibility areas of active peers should be adaptable and linked with the active peers' heterogeneity. Therefore, stronger active peers can overtake the responsibility of a bigger share of passive peers while weaker peers get a smaller share.

In the sub-field peer-specific monitoring, we presented two solutions, which can be further optimized by eradicating their weaknesses. We start with our first approach CapSearch. Its lengthy announcement operation in the worst case may be counteracted by bundling announcements and condensing the view of inner nodes. By bundling announcements and sending updated views up the tree every few seconds, if changes have happened, the costs could be further reduced and a few unnecessary write operations to the DHT could be circumvented. Possible negative side effects must be investigated. Condensing the view inner tree nodes have, could have a positive influence on the storage space needed, especially in bigger scenarios with more peers and capacity spaces. An inner node might store other inner nodes indicating that the whole sub tree rooted at this particular node contains peers' capacities. By doing so, we might introduce some additional hops needed to resolve the search query but the announce mechanism can terminate earlier and storage space can be saved. Lastly, the requirement of setting the number of capacity spaces and capacity classes at the very beginning could be softened. For this, changing the globally defined meta data, that is, the number of capacity spaces and capacity classes per space, could lead to a reconstruction of the tree. Whenever this meta data changes, each peer can locally calculate the new kd-tree and when peers announce their monitoring data to the tree's leaf nodes, the tree is built up automatically in a bottom-up manner. Alongside with CapSearch we presented PacketSkip, our second solution for monitoring peers' capacities. In future work PacketSkip's costs can be lowered and its performance further increased by applying some tweaks to the existing mechanisms. First of all, the high costs of the maintenance can be dealt with by optimizing the announcement process. Secondly, the unfair

workload share for each peer should be tackled. The element table sizes of the PacketSkip nodes are fixed, which is not optimal. In the future, the size should be changed during runtime to adapt itself dynamically according to the current needs. This is a trade-off between the workload share and increased traffic induced by PacketSkip node fluctuations. It needs to find a sweet spot between these two and an additional mechanism to adjust this parameter according to the current situation. For example, when a PacketSkip node perceives high fluctuation, the element table size should be increased, otherwise, if no fluctuations occur, the workload share is most probably bad, therefore the element table size can be decreased. Lastly, the lookup performance in the data structure of PacketSkip can be improved. This could be achieved by providing a peer caching mechanism to eliminate the DHT lookup needed to obtain the peer responsible for the next PacketSkip node. Especially when peer dynamics in the network are low, this would reduce the time complexity to $\mathcal{O}(\log N)$.

In the end, we presented WebP2P and Chunked-Swarm and raise the question about the applicability of current p2p applications. Our first solution called WebP2P can be optimized in some ways, which has been shown by our evaluation. First, delta updates or a revision system to store the buddy list will lower the required bandwidth dramatically. Furthermore, the use of web workers would reduce and most probably completely circumvent potential GUI freezes caused by heavy calculations, which are for example performed in the crypto module. Especially user with weaker CPUs would appreciate this improvement. On the side of the security, we should investigate whether route poisoning can harm our system and we should introduce an end-to-end encryption as the current security model uses hop-by-hop encryption. Lastly, we should evaluate our improved system under churn, which is an important step towards the evaluation of real-world scenarios. For our second solution, Chunked-Swarm, a lot of work has been done after we published the paper. The first steps after the paper included a lot of error handling, error correction and improving its churn resilience in order to cope with real-life scenarios. Unfortunately, further improvement has been performed behind closed doors as it is now a company secret.

We need to add, that the Research Question PS5.2 should be considered into more detail. We asked whether our solution fulfills performance requirements and demanded a comparison between the web environment and the non-web environment. We answered the first part with an analysis of our system's performance and costs. However, the last part of the question should be re-considered and a comparison between a standalone p2p application should be compared with the ported version of the same application, which runs in the web environment using WebRTC. For this, performance metrics need to be examined and an evaluation setup needs to be arranged.

7.3 Closing Words

In this dissertation we refer to monitoring as the process of gathering and disseminating data and thus offer services the opportunity to react. The word monitoring can be related with the process of being surveilled, which can cause a bad feeling in the society. It can be associated with privacy issues, which are currently discussed in society and occurred in the history. Examples are the mass surveillance revealed through the Snowden affair in 2013 and (regional) privacy groups fighting nowadays against privacy issues in big social networks. Thus, we need

to differentiate between monitoring and surveillance in our context's meaning and shaping. We define monitoring in the decentralized context as a continuous capturing of the p2p system's state and its peers' capacities. Hence, we state that it depends on what data we monitor and what actions are performed on the captured data. When monitoring peer-specific information, we consider capacities which peers want to make available for other peers to utilize. So, the peers provide information about their capabilities to support or perform calculations, depending on the service. Information about the system is gathered by system-specific monitoring approaches and it is made available to services which can react on the data. This can be described as the surveillance of the system's structure and not the users maintaining it, as services may react upon the monitored data. However, we state that we do not intend in surveilling participants of such communication networks as we neither collect sensitive user information nor react on monitoring data to surveil single users or groups of users.

The future of decentralized monitoring systems highly depends on the future of decentralized networks they operate on. In fact, we currently experience a shift of decentralized networks from the consumer market to the professional segment, which are due to the rising trends of fog and edge computing. These trends are driven by the high number of devices brought by the Internet of Things. In contrast to cloud computing, which is still dominant today, fog and edge computing pursue a decentralized approach to utilize unused resources at the edge of the Internet to process the data close to the source. By doing so, we experience a shift from a centralized processing in the cloud towards a hybrid approach in the fog and a pure decentralized processing at the edge. As parts from the well-researched field of p2p networks can be inherited for the decentralized processing at the edge, p2p techniques may receive more attention in the future. Referring to the monitoring, especially our capacity-based search can be of interest for the use-case of sensors delegating work to strong devices. Hence, our efforts in this dissertation can be helpful to realize this rising trend and findings can be re-used to drive research forward in this segment.

Personal Publications

Articles

Kalman Graffi and Andreas Disterhöft. "SkyEye: A tree-based peer-to-peer monitoring approach". In: *Elsevier Pervasive and Mobile Computing, Volume 40*. 2017.

Reviewed conference papers

Andreas Disterhöft and Kalman Graffi. "Minicamp: Middleware for Incomplete Participation in Structured Peer-to-Peer Monitoring Protocols". In: *Proceedings of the 32^{nd} IEEE International Conference on Advanced Information Networking and Applications (AINA)*. 2018.

Andreas Disterhöft, Phillip Sandkühler, Andre Ippisch and Kalman Graffi. "Mr.Tree: Multiple Realities in Tree-based Monitoring Overlays for Peer-to-Peer Networks". In: *Proceedings of the IEEE International Conference on Computing, Networking and Communications (ICNC)*. 2018.

Andreas Disterhöft and Kalman Graffi. "Minicamp: Prototype for Partial Participation in Structured Peer-to-Peer Monitoring Protocols". In: *Proceedings of the 42^{nd} IEEE International Conference on Local Computer Networks (LCN)*. 2017.

Andreas Disterhöft, Andreas Funke and Kalman Graffi. "PacketSkip: Skip Graph for Multidimensional Search in Structured Peer-to-Peer Systems". In: *Proceedings of the 11^{th} IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*. 2017.

Andreas Disterhöft and Kalman Graffi. "CapSearch: Capacity-Based Search in Highly Dynamic Peer-to-Peer Networks". In: *Proceedings of the 31^{st} IEEE International Conference on Advanced Information Networking and Applications (AINA)*. 2017.

Andreas Disterhöft and Kalman Graffi. "Convex Hull Watchdog: Mitigation of Malicious Nodes in Tree-Based P2P Monitoring Systems". In: *Proceedings of the 41^{st} IEEE International Conference on Local Computer Networks (LCN)*. 2016.

Andreas Disterhöft and Kalman Graffi. "Protected Chords in the Web: Secure P2P Framework for Decentralized Online Social Networks". In: *Proceedings of the 15^{th} IEEE International Conference on Peer-to-Peer Computing (P2P)*. 2015.

Christopher Probst, Andreas Disterhöft and Kalman Graffi. "Chunked-Swarm: Divide and Conquer for Real-time Bounds in Video Streaming". In: *Proceedings of the 15^{th} IEEE International Conference on Next Generation Wired/Wireless Advanced Networks and Systems (New2An)*. 2015.

Andreas Disterhöft and Kalman Graffi. "A Browser-based Secure P2P Framework for Decentralized Online Social Networks". In: *Proceedings of the 2nd IEEE International Conference on Networked Systems (NetSys)*. 2015.

Bibliography

[1] Harald Alvestrand. *Overview: Real Time Protocols for Browser-based Applications*. https://tools.ietf.org/html/draft-ietf-rtcweb-overview-00. Accessed: 2018-03-12. 2012 (Page: 73).

[2] Hani Alzaid, Ernest Foo, and Juan Gonzalez Nieto. "Secure Data Aggregation in Wireless Sensor Network: A Survey". In: *Proceedings of the ACM Australasian Conference on Information Security (AISC) - Volume 81*. Australian Computer Society, Inc. 2008, pp. 93–105 (Page: 48).

[3] Tobias Amft. "The Impact of Resource Sharing on Coexisting P2P Overlays and Stacked Overlay Modules". PhD thesis. Heinrich Heine University Düsseldorf, Germany, 2017. URL: https://docserv.uni-duesseldorf.de/servlets/DerivateServlet/Derivate-47062/Dissertation-Tobias-Amft-2017-1.pdf (Page: 21).

[4] Apple. *Safari 11.0: Release notes*. https://developer.apple.com/library/content/releasenotes/General/WhatsNewInSafari/Articles/Safari_11_0.html. Accessed: 2018-03-12. 2017 (Page: 75).

[5] James Aspnes and Gauri Shah. "Skip Graphs". In: *ACM Transactions on Algorithms (TALG)* 3.4 (2007) (Pages: 21, 68, 71).

[6] Stefan Axelsson. *Intrusion Detection Systems: A Survey and Taxonomy*. Technical Report: Chalmers University of Technology, Gothenburg, Sweden. Technical Report 99-15, 2000 (Page: 49).

[7] Randolph Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, and Daniel Starin. "Persona: An Online Social Network with User-Defined Privacy". In: *Proceedings of the ACM International Conference on Special Interest Group on Data Communication (SIGCOMM)*. 2009, pp. 135–146 (Page: 80).

[8] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. "Scalable Application Layer Multicast". In: *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. 2002, pp. 205–217 (Page: 85).

[9] Mark Baugher, David McGrew, Mats Naslund, Elisabetta Carrara, and Karl Norrman. *RFC 3711: The Secure Real-time Transport Protocol (SRTP)*. https://tools.ietf.org/html/rfc3711. Accessed: 2018-03-12. 2004 (Page: 74).

[10] Mayank Bawa, Hector Garcia-Molina, Aristides Gionis, and Rajeev Motwani. *Estimating Aggregates on a Peer-to-Peer Network*. Technical Report: Department for Computer Science at Stanford University InfoLab, USA, 2003 (Pages: 26, 33, 34, 39, 89).

[11] Adam Bergkvist, Daniel C. Burnett, Cullen Jennings, and Anant Narayanan. *WebRTC 1.0: Real-time Communication Between Browsers*. https://www.w3.org/TR/2011/WD-webrtc-20111027/. Accessed: 2018-03-12. 2011 (Page: 73).

[12] Ranjita Bhagwan, George Varghese, and Geoffrey M. Voelker. *Cone: Augmenting DHTs to Support Distributed Resource Discovery.* Technical Report: Department for Computer Science and Engineering at University of California, San Diego, USA. Technical Report CS2003-0755, 2003 (Pages: 33, 47, 48, 52).

[13] Ames Bielenberg, Lara Helm, Anthony Gentilucci, Dan Stefanescu, and Honggang Zhang. "The Growth of Diaspora – A Decentralized Online Social Network in the Wild". In: *Proceedings of the IEEE Infocom Computer Communications Workshop (INFOCOM WKSHPS)*. 2012, pp. 13–18 (Pages: 75, 77, 80).

[14] Francesco Blasa, Simone Cafiero, Giancarlo Fortino, and Giuseppe Di Fatta. "Symmetric Push-Sum Protocol for Decentralised Aggregation". In: *Proceedings of the International Conference on Advances in P2P Systems (AP2PS)*. 2011, pp. 27–32 (Pages: 19, 26, 31, 34, 88).

[15] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. "Fog Computing and Its Role in the Internet of Things". In: *Proceedings of the ACM Workshop on Mobile Cloud Computing (MCC)*. 2012, pp. 13–16 (Page: 3).

[16] Sonja Buchegger and Anwitaman Datta. "A Case for P2P Infrastructure for Social Networks - Opportunities & Challenges". In: *Proceedings of the IEEE International Conference on Wireless On-Demand Network Systems and Services (WONS)*. 2009, pp. 161–168 (Page: 75).

[17] Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. "PeerSoN: P2P Social Networking — Early Experiences and Insights". In: *Proceedings of the ACM EuroSys Workshop on Social Network Systems (SNS)*. 2009, pp. 46–52 (Pages: 75, 77, 81).

[18] Emanuele Carlini, Alessandro Lulli, and Laura Ricci. "DRAGON: Multidimensional range queries on distributed aggregation trees". In: *Elsevier Future Generation Computer Systems* 55 (2016), pp. 101–115 (Page: 66).

[19] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. "SplitStream: High-Bandwidth Multicast in Cooperative Environments". In: *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*. 2003, pp. 298–313 (Pages: 39, 85).

[20] Haowen Chan, Adrian Perrig, and Dawn Xiaodong Song. "Secure Hierarchical In-Network Aggregation in Sensor Networks". In: *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. 2006, pp. 278–287 (Page: 48).

[21] Tammy Kam-Hung Chan, Shueng-Han Gary Chan, and Ali C. Begen. "SPANC: Optimizing Scheduling Delay for Peer-to-Peer Live Streaming". In: *IEEE Transactions on Multimedia* 12.7 (2010), pp. 743–753 (Page: 86).

[22] David Clark. "The Design Philosophy of the DARPA Internet Protocols". In: *ACM SIGCOMM Computer Communication Review* 18.4 (1988), pp. 106–114 (Page: 1).

[23] Jeffrey Considine, Feifei Li, George Kollios, and John Byers. "Approximate Aggregation Techniques for Sensor Databases". In: *Proceedings of the IEEE International Conference on Data Engineering*. 2004, pp. 449–460 (Page: 39).

[24] Leucio Antonio Cutillo, Refik Molva, and Melek Önen. "Safebook: A Distributed Privacy Preserving Online Social Network". In: *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM)*. 2011, pp. 1–3 (Page: 80).

[25] Frank Dabek, Ben Y. Zhao, Peter Druschel, John Kubiatowicz, and Ion Stoica. "Towards a Common API for Structured Peer-to-Peer Overlays". In: *Proceedings of the USENIX International Workshop on Peer-to-Peer Systems (IPTPS)*. 2003, pp. 33–44 (Pages: 27, 28, 35, 54).

[26] Alberto Dainotti, Antonio Pescapè, and Kimberly C. Claffy. "Issues and Future Directions in Traffic Classification". In: *IEEE Network* 26.1 (2012), pp. 35–40 (Page: 2).

[27] Mads Dam and Rolf Stadler. "A Generic Protocol for Network State Aggregation". In: *Proceedings of the Radiovetenskap och Kommunikation (RVK)*. 2005 (Pages: 26, 32, 35, 36, 38, 48).

[28] Vasilios Darlagiannis. "Overlay Network Mechanisms for Peer-to-Peer Systems". PhD thesis. Technische Universität Darmstadt, Germany, 2005. URL: http://elib.tu-darmstadt.de/diss/000699 (Page: 60).

[29] Anwitaman Datta and Rajesh Sharma. "GoDisco: Selective Gossip Based Dissemination of Information in Social Community Based Overlays". In: *Proceedings of the IEEE International Conference on Distributed Computing and Networking (ICDCN)*. 2011, pp. 227–238 (Page: 80).

[30] Anwitaman Datta, Sonja Buchegger, Le-Hung Vu, Thorsten Strufe, and Krzysztof Rzadca. "Decentralized Online Social Networks". In: *Handbook of Social Network Technologies and Applications*. 2010, pp. 349–378 (Page: 75).

[31] Tim Dierks and Eric Rescorla. *RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2*. https://tools.ietf.org/html/rfc5246. Accessed: 2018-03-12. 2008 (Page: 74).

[32] Andreas Disterhöft, Andreas Funke, and Kalman Graffi. "PacketSkip: Skip Graph for Multidimensional Search in Structured Peer-to-Peer Systems". In: *Proceedings of the IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*. 2017, pp. 21–30 (Pages: 21, 23, 60, 61, 67, 68, 92).

[33] Andreas Disterhöft and Kalman Graffi. "A Browser-based Secure P2P Framework for Decentralized Online Social Networks". In: *Proceedings of the IEEE International Conference on Networked Systems (NetSys)*. 2015 (Pages: 22, 23, 76, 77, 92).

[34] Andreas Disterhöft and Kalman Graffi. "CapSearch: Capacity-Based Search in Highly Dynamic Peer-to-Peer Networks". In: *Proceedings of the IEEE International Conference on Advanced Information Networking and Applications (AINA)*. 2017, pp. 621–630 (Pages: 21, 23, 60, 61, 63, 67, 68, 71, 91).

[35] Andreas Disterhöft and Kalman Graffi. "Convex Hull Watchdog: Mitigation of Malicious Nodes in Tree-Based P2P Monitoring Systems". In: *Proceedings of the IEEE International Conference on Local Computer Networks (LCN)*. 2016, pp. 52–60 (Pages: 20, 23, 28, 46, 48, 90).

[36] Andreas Disterhöft and Kalman Graffi. "Minicamp: Middleware for Incomplete Participation in Structured Peer-to-Peer Monitoring Protocols". In: *Proceedings of the IEEE International Conference on Advanced Information Networking and Applications (AINA)*. 2018 (Pages: 20, 21, 23, 56, 90).

[37] Andreas Disterhöft and Kalman Graffi. "Minicamp: Prototype for Partial Participation in Structured Peer-to-Peer Monitoring Protocols". In: *Proceedings of the IEEE International Conference on Local Computer Networks (LCN)*. 2017, pp. 203–206 (Pages: 20, 21, 23, 28, 53, 56, 90).

[38] Andreas Disterhöft and Kalman Graffi. "Protected Chords in the Web: Secure P2P Framework for Decentralized Online Social Networks". In: *Proceedings of the IEEE International Conference on Peer-to-Peer Computing (P2P)*. 2015, pp. 1–5 (Pages: 22, 23, 76, 79, 92).

[39] Andreas Disterhöft, Phillip Sandkühler, Andre Ippisch, and Kalman Graffi. "Mr.Tree: Multiple Realities in Tree-based Monitoring Overlays for Peer-to-Peer Networks". In: *Proceedings of the IEEE International Conference on Computing, Networking and Communications (ICNC)*. 2018 (Pages: 19, 20, 23, 26, 28, 34, 35, 37, 41, 43, 89).

[40] Prasanna Ganesan, Beverly Yang, and Hector Garcia-Molina. "One Torus to Rule them All: Multi-dimensional Queries in P2P Systems". In: *Proceedings of the ACM International Workshop on the Web and Databases (WebDB)*. 2004, pp. 19–24 (Pages: 65, 72).

[41] Jun Gao and Peter Steenkiste. "Efficient Support for Similarity Searches in DHT-based Peer-to-Peer Systems". In: *Proceedings of the IEEE International Conference on Communications (ICC)*. 2007, pp. 1867–1874 (Page: 65).

[42] Norbert Goebel, Tobias Krauthoff, Kalman Graffi, and Martin Mauve. "Moving measurements: Measuring network characteristics of mobile cellular networks on the move". In: *Elsevier Computer Communications* 97 (2017), pp. 81–95 (Page: 54).

[43] Jennifer Golbeck and Matthew Rothstein. "Linking Social Networks on the Web with FOAF: A Semantic Web Case Study". In: *Proceedings of the Conference on Artificial Intelligence (AAAI)*. 2008, pp. 1138–1143 (Page: 80).

[44] Kalman Graffi. "PeerfactSim.KOM: A P2P System Simulator - Experiences and Lessons Learned". In: *Proceedings of the IEEE International Conference on Peer-to-Peer Computing (P2P)*. 2011, pp. 154–155 (Pages: 19, 31).

[45] Kalman Graffi and Andreas Disterhöft. "SkyEye: A tree-based peer-to-peer monitoring approach". In: *Elsevier Pervasive and Mobile Computing* 40 (2017), pp. 593–610 (Pages: 19, 23, 26–28, 35, 38, 41, 47, 48, 52, 88).

[46] Kalman Graffi, Christian Gross, Dominik Stingl, Daniel Hartung, Aleksandra Kovacevic, and Ralf Steinmetz. "LifeSocial.KOM: A Secure and P2P-based Solution for Online Social Networks". In: *Proceedings of the IEEE International Conference on Consumer Communications and Networking Conference (CCNC)*. 2011, pp. 554–558 (Pages: 25, 75, 77, 81).

[47] Kalman Graffi, Sebastian Kaune, Konstantin Pussep, Aleksandra Kovacevic, and Ralf Steinmetz. "Load Balancing for Multimedia Streaming in Heterogeneous Peer-to-Peer Systems". In: *Proceedings of the ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*. 2008, pp. 99–104 (Page: 84).

[48] Kalman Graffi, Dominik Stingl, Julius Rückert, and Aleksandra Kovacevic. "Monitoring and Management of Structured Peer-to-Peer Systems". In: *Proceedings of the IEEE International Conference on Peer-to-Peer Computing (P2P)*. 2009, pp. 311–320 (Pages: 45, 51).

[49] Kalman Graffi, Patrick Mukherjee, Burkhard Menges, Daniel Hartung, Aleksandra Kovacevic, and Ralf Steinmetz. "Practical Security in P2P-based Social Networks". In: *Proceedings of the IEEE International Conference on Local Computer Networks (LCN)*. 2009, pp. 269–272 (Page: 78).

[50] 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 IEEE International Conference on Parallel and Distributed Systems (ICPADS)*. 2008, pp. 279–286 (Pages: 26, 64).

[51] Kalman Graffi, Christian Groß, Dominik Stingl, Hoang Nguyen, Aleksandra Kovacevic, and Ralf Steinmetz. "Towards a P2P Cloud: Reliable Resource Reservations in Unreliable P2P Systems". In: *Proceedings of the IEEE International Conference on Parallel and Distributed Systems (ICPADS)*. 2010, pp. 27–34 (Pages: 6, 7, 54, 60, 64).

[52] Kálmán György Graffi. "Monitoring and Management of Peer-to-Peer Systems". PhD thesis. Technische Universität Darmstadt, Germany, 2010. URL: http://tuprints.ulb.tu-darmstadt.de/2248/ (Pages: 9, 19, 34, 88).

[53] Ilya Grigorik. *High Performance Browser Networking: What every web developer should know about networking and web performance*. Accessed: 2018-03-12. O'Reilly Media, Inc., 2013. URL: https://hpbn.co/ (Page: 74).

[54] Saikat Guha, Neil Daswani, and Ravi Jain. "An Experimental Study of the Skype Peer-to-Peer VoIP System". In: *Proceedings of the USENIX International Workshop on Peer-to-Peer Systems (IPTPS)*. 2006, pp. 27–34 (Pages: 79, 80, 93).

[55] Dominique Guinard, Vlad Trifa, Stamatis Karnouskos, Patrik Spiess, and Domnic Savio. "Interacting with the SOA-based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services". In: *IEEE Transactions on Services Computing* 3.3 (2010), pp. 223–235 (Page: 60).

[56] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W. Ross. "A Measurement Study of a Large-Scale P2P IPTV System". In: *IEEE Transactions on Multimedia* 9.8 (2007), pp. 1672–1687 (Page: 2).

[57] Ian Hickson, Robin Berjon, Steve Faulkner, Travis Leithead, Erika Doyle Navara, Edward O'Connor, and Silvia Pfeiffer. *HTML5: Hypertext Markup Language (HTML) Version 5*. https://www.w3.org/TR/html5/. Accessed: 2018-03-12. 2017 (Page: 73).

[58] Tobias Hoßfeld, Sebastian Egger, Raimund Schatz, Markus Fiedler, Kathrin Masuch, and Charlott Lorentzen. "Initial Delay vs. Interruptions: Between the Devil and the Deep Blue Sea". In: *Proceedings of the IEEE International Workshop on Quality of Multimedia Experience (QoMEX)*. 2012, pp. 1–6 (Page: 25).

[59] Alfonso Iacovazzi and Yuval Elovici. "Network Flow Watermarking: A Survey". In: *IEEE Communications Surveys & Tutorials* 19.1 (2017), pp. 512–530 (Page: 2).

[60] Boris Iglewicz and David Caster Hoaglin. *How to Detect and Handle Outliers*. Vol. 16. ASQC Quality Press, 1993 (Page: 47).

[61] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. "Networking Named Content". In: *Proceedings of the ACM International Conference on Emerging Networking Experiments and Technology (CoNEXT)*. 2009, pp. 1–12 (Page: 3).

[62] Hosagrahar V. Jagadish, Beng Chin Ooi, Martin Rinard, and Quang Hieu Vu. "BATON: A Balanced Tree Structure for Peer-to-Peer Networks". In: *Proceedings of the ACM International Conference on Very Large Data Bases (VLDB)*. 2005, pp. 661–672 (Page: 65).

[63] Hosagrahar V. Jagadish, Beng Chin Ooi, Kian-Lee Tan, Quang Hieu Vu, and Rong Zhang. "Speeding up Search in Peer-to-Peer Networks with A Multi-way Tree Structure". In: *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. 2006, pp. 1–12 (Page: 65).

[64] Hosagrahar V. Jagadish, Beng Chin Ooi, Quang Hieu Vu, Rong Zhang, and Aoying Zhou. "VBI-Tree: A Peer-to-Peer Framework for Supporting Multi-Dimensional Indexing Schemes". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 2006, p. 34 (Page: 65).

[65] Márk Jelasity, Alberto Montresor, and Özalp Babaoglu. "T-Man: Gossip-based Fast Overlay Topology Construction". In: *Elsevier Computer Networks* 53.13 (2009), pp. 2321–2339 (Pages: 26, 34).

[66] Paulo Jesus, Carlos Baquero, and Paulo Sérgio Almeida. "A Survey of Distributed Data Aggregation Algorithms". In: *IEEE Communications Surveys & Tutorials* 17.1 (2015), pp. 381–404 (Pages: 5, 36, 38).

[67] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. "The EigenTrust Algorithm for Reputation Management in P2P Networks". In: *Proceedings of the ACM International Conference on World Wide Web (WWW)*. 2003, pp. 640–651 (Pages: 11, 49).

[68] Saeed Kargar and Leyli Mohammad-Khanli. "Fractal: An advanced multidimensional range query lookup protocol on nested rings for distributed systems". In: *Elsevier Journal of Network and Computer Applications* 87 (2017), pp. 147 –168 (Page: 64).

[69] David Kempe, Alin Dobra, and Johannes Gehrke. "Gossip-Based Computation of Aggregate Information". In: *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*. 2003, pp. 482–491 (Pages: 26, 34).

[70] John Lazzaro. *RFC 4571: Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport*. `https://tools.ietf.org/html/rfc4571`. Accessed: 2018-03-12. 2006 (Page: 74).

[71] Jure Leskovec and Eric Horvitz. "Planetary-Scale Views on a Large Instant-Messaging Network". In: *Proceedings of the ACM International Conference on World Wide Web (WWW)*. 2008, pp. 915–924 (Pages: 79, 80, 93).

[72] Ji Li, Karen Sollins, and Dah-Yoh Lim. "Implementing Aggregation and Broadcast over Distributed Hash Tables". In: *ACM SIGCOMM Computer Communication Review* 35.1 (2005), pp. 81–92 (Page: 39).

[73] Wei Liang, Jingping Bi, Rong Wu, Zhenyu Li, and Chen Li. "On Characterizing PPStream: Measurement and Analysis of P2P IPTV under Large-Scale Broadcasting". In: *Proceedings of the IEEE International Conference on Global Telecommunications (GLOBECOM)*. 2009, pp. 1–6 (Page: 2).

[74] Zhengye Liu, Yanming Shen, Keith W. Ross, Shivendra S. Panwar, and Yao Wang. "LayerP2P: Using Layered Video Chunks in P2P Live Streaming". In: *IEEE Transactions on Multimedia* 11.7 (2009), pp. 1340–1352 (Page: 85).

[75] Zhengye Liu, Yanming Shen, Keith W. Ross, Shivendra S. Panwar, and Yao Wang. "Substream Trading: Towards an open P2P Live Streaming System". In: *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*. 2008, pp. 94–103 (Page: 86).

[76] Scott Ludwig, Joe Beda, Peter Saint-Andre, Robert McQueen, Sean Egan, and Joe Hildebrand. *XEP-0166: Jingle*. https://xmpp.org/extensions/xep-0166.html. Accessed: 2018-03-12. 2016 (Page: 74).

[77] Rohan Mahy, Philip Matthews, and Jonathan Rosenberg. *RFC 5766: Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)*. https://tools.ietf.org/html/rfc5766. Accessed: 2018-03-12. 2010 (Page: 74).

[78] Gregor Maier, Anja Feldmann, Vern Paxson, and Mark Allman. "On Dominant Characteristics of Residential Broadband Internet Traffic". In: *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC)*. 2009, pp. 90–102 (Page: 2).

[79] Rafik Makhloufi, Guillaume Doyen, Grégory Bonnet, and Dominique Gaïti. "A survey and performance evaluation of decentralized aggregation schemes for autonomic management". In: *Wiley International Journal of Network Management* 24.6 (2014), pp. 469–498 (Pages: 5, 26, 28, 30, 32, 36, 38).

[80] Rafik Makhloufi, Grégory Bonnet, Guillaume Doyen, and Dominique Gaïti. "Decentralized aggregation protocols in Peer-to-Peer networks: a survey". In: *Proceedings of the IEEE International Workshop on Modelling Autonomic Communications Environments (MACE)*. 2009, pp. 111–116 (Page: 33).

[81] Sergio Marti and Hector Garcia-Molina. "Taxonomy of trust: Categorizing P2P reputation systems". In: *Elsevier Computer Networks* 50.4 (2006), pp. 472–484 (Page: 49).

[82] Giuliano Mega, Alberto Montresor, and Gian Pietro Picco. "Efficient Dissemination in Decentralized Social Networks". In: *Proceedings of the IEEE International Conference on Peer-to-Peer Computing (P2P)*. 2011, pp. 338–347 (Page: 80).

[83] Parag S. Mogre, Kalman Graffi, Matthias Hollick, and Ralf Steinmetz. "AntSec, WatchAnt, and AntRep: Innovative Security Mechanisms for Wireless Mesh Networks". In: *Proceedings of the IEEE International Conference on Local Computer Networks (LCN)*. 2007, pp. 539–547 (Page: 49).

[84] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. http://bitcoin.org/. Accessed: 2018-03-23 (Page: 1).

[85] *NetMarketShare: Operating Systems' Share*. https://www.netmarketshare.com/operating-system-market-share.aspx. Accessed: 2018-03-12 (Page: 51).

[86] David Oppenheimer, Brent Chun, David Patterson, Alex C. Snoeren, and Amin Vahdat. "Service Placement in a Shared Wide-Area Platform". In: *Proceedings of the USENIX Annual Technical Conference*. 2006, pp. 273–288 (Page: 60).

[87] Venkata N. Padmanabhan, Helen J. Wang, Philip A. Chou, and Kunwadee Sripanidkulchai. "Distributing Streaming Media Content Using Cooperative Networking". In: *Proceedings of the ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*. 2002, pp. 177–186 (Page: 84).

[88] Vinay S. Pai, Kapil Kumar, Karthik Tamilmani, Vinay Sambamurthy, and Alexander E. Mohr. "Chainsaw: Eliminating Trees from Overlay Multicast". In: *Proceedings of the USENIX International Workshop on Peer-to-Peer Systems (IPTPS)*. 2005, pp. 127–140 (Page: 85).

[89] Andrea Passarella. "A survey on content-centric technologies for the current Internet: CDN and P2P solutions". In: *Elsevier Computer Communications* 35.1 (2012), pp. 1–32 (Page: 3).

[90] Thomas Paul, Benjamin Greschbach, Sonja Buchegger, and Thorsten Strufe. "Exploring Decentralization Dimensions of Social Networking Services: Adversaries and Availability". In: *Proceedings of the ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research (HotSocial)*. 2012, pp. 49–56 (Page: 75).

[91] Dimitrios Pendarakis, Sherlia Shi, Dinesh Verma, and Marcel Waldvogel. "ALMI: An Application Level Multicast Infrastructure". In: *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS)*. 2001, pp. 49–60 (Page: 84).

[92] Diego Perino and Matteo Varvello. "A Reality Check for Content Centric Networking". In: *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking (ICN)*. 2011, pp. 44–49 (Page: 3).

[93] Marcin Pietrzyk, Jean-Laurent Costeux, Guillaume Urvoy-Keller, and Taoufik En-Najjary. "Challenging Statistical Classification for Operational Usage: the ADSL Case". In: *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC)*. 2009, pp. 122–135 (Page: 2).

[94] John Postel. *RFC 768: User Datagram Protocol*. https://www.ietf.org/rfc/rfc768.txt. Accessed: 2018-03-12. 1980 (Page: 74).

[95] John Postel. *RFC 793: Transmission Control Protocol*. https://tools.ietf.org/html/rfc793. Accessed: 2018-03-12. 1981 (Page: 74).

[96] Alberto Gonzalez Prieto and Rolf Stadler. "A-GAP: An Adaptive Protocol for Continuous Network Monitoring with Accuracy Objectives". In: *IEEE Transactions on Network and Service Management (TNSM)* 4.1 (2007), pp. 2–12 (Pages: 26, 32, 35, 48).

[97] Christopher Probst, Andreas Disterhöft, and Kalman Graffi. "Chunked-Swarm: Divide and Conquer for Real-time Bounds in Video Streaming". In: *Proceedings of the IEEE International Conference on Next Generation Wired/Wireless Advanced Networks and Systems (New2An)*. 2015, pp. 198–210 (Pages: 22, 23, 76, 82, 85, 93).

[98] Christopher Probst and Alexander Schäfer. *StriveCDN: The Power of WebRTC Live Streaming*. https://strivecdn.com/. Accessed: 2018-03-12. 2017 (Pages: 76, 82).

[99] Vitaliy Rapp and Kalman Graffi. "Continuous Gossip-based Aggregation Through Dynamic Information Aging". In: *Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN)*. 2013, pp. 1–7 (Page: 34).

[100] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. "A Scalable Content-Addressable Network". In: *Proceedings of the ACM SIGCOMM International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. 2001, pp. 161–172 (Page: 65).

[101] Eric Rescorla and Nagendra Modadugu. *RFC 6347: Datagram Transport Layer Security Version 1.2*. https://tools.ietf.org/html/rfc6347. Accessed: 2018-03-12. 2012 (Page: 74).

[102] Jonathan Rosenberg. *RFC 5245: Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols*. https://tools.ietf.org/html/rfc5245. Accessed: 2018-03-12. 2010 (Page: 74).

[103] Jonathan Rosenberg, Rohan Mahy, Philip Matthews, and Dan Wing. *RFC 5389: Session Traversal Utilities for NAT (STUN)*. https://tools.ietf.org/html/rfc5389. Accessed: 2018-03-12. 2008 (Page: 74).

[104] Antony Rowstron and Peter Druschel. "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems". In: *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*. 2001, pp. 329–350 (Pages: 25, 27, 35, 57, 60).

[105] Antony Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. "SCRIBE: The Design of a Large-Scale Event Notification Infrastructure". In: *Proceedings of the International Workshop on Networked Group Communication (NGC)*. 2001, pp. 30–43 (Page: 39).

[106] Julius Rückert, Tamara Knierim, and David Hausheer. "Clubbing with the Peers: A Measurement Study of BitTorrent Live". In: *Proceedings of the IEEE International Conference on Peer-to-Peer Computing (P2P)*. 2014, pp. 1–10 (Page: 85).

[107] Nikita Salnikov-Tarnovski. *Plumbr Java Version Share*. `https://plumbr.eu/blog/java/java-version-and-vendor-data-analyzed-2016-edition`. Accessed: 2018-03-12 (Page: 51).

[108] Marc Sanchez-Artigas, Pedro Garcia Lopez, and Antonio Skarmeta. "DECA: A Hierarchical Framework for DECentralized Aggregation in DHTs". In: *Proceedings of the International Workshop on Distributed Systems: Operations and Management (DSOM)*. 2006, pp. 246–257 (Page: 39).

[109] Sakir Sezer, Sandra Scott-Hayward, Pushpinder Kaur Chouhan, Barbara Fraser, David Lake, Jim Finnegan, Niel Viljoen, Marc Miller, and Navneet Rao. "Are we ready for SDN? - Implementation Challenges for Software-Defined Networks". In: *IEEE Communications Magazine* 51.7 (2013), pp. 36–43 (Page: 3).

[110] Rajesh Sharma and Anwitaman Datta. "SuperNova: Super-peers Based Architecture for Decentralized Online Social Networks". In: *Proceedings of the IEEE International Conference on Communication Systems and Networks (COMSNETS)*. 2012, pp. 1–10 (Page: 81).

[111] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. "Edge Computing: Vision and Challenges". In: *IEEE Internet of Things Journal* 3.5 (2016), pp. 637–646 (Page: 3).

[112] Yanfeng Shu, Beng Chin Ooi, Kian-Lee Tan, and Aoying Zhou. "Supporting Multi-dimensional Range Queries in Peer-to-Peer Systems". In: *Proceedings of the IEEE International Conference on Peer-to-Peer Computing (P2P)*. 2005, pp. 173–180 (Page: 72).

[113] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. *Analyzing Peer Behavior in KAD*. Technical Report: Department for Corporate Communications at Institut Eurecom, Biot, France. Research Report RR-07-205, 2007 (Pages: 31, 88).

[114] Randall Stewart. *RFC 4960: Stream Control Transmission Protocol*. `https://tools.ietf.org/html/rfc4960`. Accessed: 2018-03-12. 2007 (Page: 74).

[115] Ion Stoica, Robert Morris, David Liben-Nowell, David Karger, Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications". In: *IEEE/ACM Transactions on Networking (TON)* 11.1 (2003), pp. 17–32 (Pages: 27, 31, 35, 57, 60).

[116] Yuzhe Tang, Jianliang Xu, Shuigeng Zhou, and Wang-chien Lee. "m-LIGHT: Indexing Multi-Dimensional Data over DHTs". In: *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*. 2009, pp. 191–198 (Page: 65).

[117] Ruixiong Tian, Yongqiang Xiong, Qian Zhang, Bo Li, Ben Y. Zhao, and Xing Li. "Hybrid Overlay Structure Based on Random Walks". In: *Proceedings of the USENIX International Workshop on Peer-to-Peer Systems (IPTPS)*. 2005, pp. 152–162 (Page: 85).

[118] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. "A Survey of DHT Security Techniques". In: *ACM Computing Surveys (CSUR)* 43.2 (2011) (Page: 45).

[119] Robbert Van Renesse, Kenneth P. Birman, and Werner Vogels. "Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining". In: *ACM Transactions on Computer Systems (TOCS)* 21.2 (2003), pp. 164–206 (Page: 32).

[120] Dan S. Wallach. "A Survey of Peer-to-Peer Security Issues". In: *Proceedings of the International Symposium of Software Security—Theories and Systems (ISSS)*. 2003, pp. 42–57 (Page: 45).

[121] Feng Wang, Yongqiang Xiong, and Jiangchuan Liu. "mTreebone: A Collaborative Tree-Mesh Overlay Network for Multicast Video Streaming". In: *IEEE Transactions on Parallel and Distributed Systems* 21.3 (2010), pp. 379–392 (Pages: 39, 85).

[122] *WebRTC: Browser support grid*. `http://iswebrtcreadyyet.com/legacy.html`. Accessed: 2018-03-13 (Page: 75).

[123] *WebRTC: Browser support simplified*. `http://iswebrtcreadyyet.com/`. Accessed: 2018-03-13 (Page: 75).

[124] Philipp Wette and Kalman Graffi. "Adding Capacity-Aware Storage Indirection to Homogeneous Distributed Hash Tables". In: *Proceedings of the IEEE International Conference on Networked Systems (NetSys)*. 2013, pp. 35–42 (Page: 64).

[125] David Isaac Wolinsky, Pierre St. Juste, P. Oscar Boykin, and Renato Figueiredo. "OverSoc: Social Profile Based Overlays". In: *Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE)*. 2010, pp. 205–210 (Page: 81).

[126] Fetahi Wuhib, Mads Dam, and Rolf Stadler. "Decentralized detection of global threshold crossings using aggregation trees". In: *Elsevier Computer Networks* 52.9 (2008), pp. 1745–1761 (Pages: 26, 32, 35, 48).

[127] Fetahi Wuhib and Rolf Stadler. *M-GAP–A New Pattern for cfengine and other distributed software*. Technical Report: Department for Network and Systems Engineering at KTH Royal Institute of Technology, Stockholm, Sweden. Technical Report 2.10, 2007 (Pages: 26, 32, 33).

[128] Fetahi Wuhib, Mads Dam, Rolf Stadler, and Alexander Clem. "Robust Monitoring of Network-Wide Aggregates Through Gossiping". In: *IEEE Transactions on Network and Service Management* 6.2 (2009), pp. 95–109 (Page: 34).

[129] Susu Xie, Bo Li, Gabriel Yik Keung, and Xinyan Zhang. "Coolstreaming: Design, Theory, and Practice". In: *IEEE Transactions on Multimedia* 9.8 (2007), pp. 1661–1671 (Page: 86).

[130] Ming Xu, Shuigeng Zhou, and Jihong Guan. "A new and effective hierarchical overlay structure for Peer-to-Peer networks". In: *Elsevier Computer Communications* 34.7 (2011), pp. 862–874 (Page: 72).

[131] Praveen Yalagandula and Mike Dahlin. "A Scalable Distributed Information Management System". In: *Proceedings of the ACM SIGCOMM International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. 2004, pp. 379–390 (Pages: 26, 32, 35, 38, 47, 48, 52).

[132] Yi Yang, Xinran Wang, Sencun Zhu, and Guohong Cao. "SDAP: A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks". In: *ACM Transactions on Information and System Security (TISSEC)* 11.4 (2008) (Page: 48).

[133] Chi Zhang, Arvind Krishnamurthy, and Randolph Y Wang. *SkipIndex: Towards a Scalable Peer-to-Peer Index Service for High Dimensional Data*. Technical Report: Department for Computer Science at Princeton University, New Jersey, USA. Technical Report 703–04, 2004 (Pages: 65, 72).

[134] Chong Zhang, Weidong Xiao, Daquan Tang, and Jiuyang Tang. "P2P-based multidimensional indexing methods: A survey". In: *Elsevier Journal of Systems and Software* 84.12 (2011), pp. 2348 –2362 (Pages: 7, 59, 60, 65).

[135] Cong Zhang and Jiangchuan Liu. "On Crowdsourced Interactive Live Streaming: A Twitch.TV-Based Measurement Study". In: *Proceedings of the ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*. 2015, pp. 55–60 (Pages: 76, 83).

[136] Xiangyang Zhang and Hossam S. Hassanein. "TreeClimber: A Network-Driven Push-Pull Hybrid Scheme for Peer-to-Peer Video Live Streaming". In: *Proceedings of the IEEE International Conference on Local Computer Networks (LCN)*. 2010, pp. 368–371 (Page: 85).

[137] Zheng Zhang, Shuming Shi, and Jing Zhu. "SOMO: Self-Organized Metadata Overlay for Resource Management in P2P DHT". In: *Proceedings of the USENIX International Workshop on Peer-to-Peer Systems (IPTPS)*. 2003, pp. 170–182 (Pages: 26, 32, 35, 47, 48, 52).

[138] Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John D. Kubiatowicz. "Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination". In: *Proceedings of the ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*. 2001, pp. 11–20 (Page: 85).

Eidesstattliche Erklärung
laut §5 der Promotionsordnung vom 06.12.2013

Ich versichere an Eides Statt, dass die Dissertation von mir selbständig und ohne unzulässige fremde Hilfe unter Beachtung der „Grundsätze zur Sicherung guter wissenschaftlicher Praxis an der Heinrich-Heine-Universität Düsseldorf“ erstellt worden ist.

Ort, Datum	Andreas Disterhöft