DISS. ETH NO. 17731
TIK-Schriftenreihe Nr. 96

Dynamics and Cooperation
Algorithmic Challenges in Peer-to-Peer Computing

A dissertation submitted to the

SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH

for the degree of

Doctor of Sciences

presented by

STEFAN SCHMID

MSc in Computer Science, ETH Zürich

born 31.07.1978

citizen of
Hedingen ZH

accepted on the recommendation of

Prof. Dr. Roger Wattenhofer, ETH Zurich, examiner
Prof. Dr. Boaz Patt-Shamir, Tel Aviv University, co-examiner
Prof. Dr. Tim Roughgarden, Stanford University, co-examiner

2008

Abstract

Peer-to-peer (p2p) computing is one of the most intriguing new networking paradigms of the last ten years, and many applications today use peer-to-peer technology, e.g. for large distributed computations, for file sharing, or for live media streaming. At the heart of the paradigm lies the idea of leveraging the resources of the system's participants. Thus, potentially scalable and robust architectures can be built.

However, making use of the decentralized resources is challenging. The peers are under the control of the individual users who may only connect to the network for a short period of time. Consequently, there are frequent membership changes and p2p systems are highly dynamic. In addition to regular joins and leaves, the participating machines (often unreliable desktops) may crash. Peer-to-peer solutions are also faced with the fact that it is not always in the (anonymous) users' interest to contribute their resources. Rather, a user may be selfish and seek to exploit the system without reciprocating.

This dissertation studies the challenges of the dynamics in p2p computing and of cooperation. We describe a system which is based on a hypercubic topology and which applies algorithms that maintain desirable network properties despite worst-case joins and leaves; these algorithms can also be used for pancake graphs. Besides membership dynamics, we investigate dynamic changes of the available bandwidth between two peers, and we analyze the throughput of different transfer protocols. In order to emphasize the importance of the cooperation challenge, we conduct a case study of BitTorrent—one of the most traffic intensive applications on the Internet—, and show that today's peer-to-peer networks still fail to fend off uncooperative peers. A game-theoretic analysis of a p2p network creation game is presented which estimates the impact of selfish behavior. We find that both the performance and the stability of a system can suffer severely. In addition, this dissertation introduces a mathematical framework which allows us to evaluate a system's robustness to malicious attacks; the framework is also useful for the analysis of social networks. The theoretic findings are complemented by a case study which identifies vulnerabilities in the popular Kad network.

Zusammenfassung

Peer-to-peer Computing ist eines der faszinierensten Paradigmen im Networking Bereich des letzten Jahrzehnts. Bereits gibt es zahlreiche Anwendungen, beispielsweise die Berechnung schwieriger mathematischer Probleme, der Austausch grosser Dateien oder live Übertragungen von Sportveranstaltungen. Dem Paradigma liegt die Idee zugrunde, dass durch Ausnutzung der Resourcen der teilnehmenden Rechner potentiell skalierbare und robuste Architekturen möglich werden.

Die Abhängigkeit von den dezentralen Resourcen der Teilnehmer bringt aber auch gewisse Schwierigkeiten mit sich. Benutzer verbinden sich oft nur für kurze Zeit zum Netzwerk, weshalb das System oft sehr dynamisch ist. Meist sind die beteiligten Maschinen auch unzuverlässig und können jederzeit ausfallen. Des weiteren ist es häufig nicht im Interesse eines (anonymen) Benutzers, selber Resourcen zum System beizutragen.

Diese Dissertation befasst sich mit den Herausforderungen, die sich aus der Dynamik und aus der Notwendigkeit zur Kooperation ergeben. Insbesondere werden wir ein auf einem Hyperwürfel basierendes System vorstellen, das trotz ununterbrochenen und schlimmstmöglichen Topologieänderungen nützliche Eigenschaften sicherstellt. Unsere Algorithmen können auch auf andere Topologien angewandt werden, beispielsweise auf Pancake Graphen. Neben der sogenannten Knotendynamik betrachten wir auch dynamische Änderungen der verfügbaren Bandbreite zwischen zwei Peers und analysieren die Effizienz verschiedener Übertragungsprotokolle. Unsere Fallstudie in BitTorrent – eine der beliebtesten Anwendungen im Internet – zeigt auf, dass heutige peer-to-peer Systeme noch keine wirksamen Mechanismen gegen unkooperatives Verhalten anwenden. Mittels einer spieltheoretische Analyse der Topologien unstrukturierter peer-to-peer Netzwerke kommen wir zum Schluss, dass ein strategisches Verhalten aber einen wesentlichen Einfluss auf die Effizienz eines Systems haben kann. Diese Arbeit führt ausserdem einen mathematischen Formalismus ein, der es ermöglicht, den potentiellen Schaden durch böswillige Teilnehmer abzuschätzen; dieser Formalismus ist auch bei der Analyse von sozialen Netzwerken einsetzbar. Die theoretischen Erkenntnisse werden ergänzt durch ein Fallbeispiel, welches Schwachstellen in Kad, einem populären peer-to-peer Netzwerk, identifiziert.

Acknowledgements

Many people have contributed to this dissertation. First of all, I am grateful to my advisor Roger Wattenhofer for his support, and for giving me the opportunity to pursue a number of different and exciting projects during my PhD time. I benefitted from his excellent knowledge—and critical view—of past and current developments in the field, and I feel lucky that the European Space Agency chose another candidate three years ago!

I am honored that two well-known researchers, Boaz Patt-Shamir and Tim Roughgarden, agreed to read through my dissertation and to serve on my committee board. I would like to acknowledge their work and thank for their comments which helped to improve the thesis.

Parts of this thesis have been presented on various occasions. I am obliged to the following people for their invitations and their feedback: Aaron Zollinger (UC Berkeley), David Parkes (Harvard University), Bobby Kleinberg and Emin Gün Sirer (Cornell University), Stephan Eidenbenz (Los Alamos National Laboratories), and Christian Scheideler (TU München). In particular, I would like to thank Sven Kosub for pointing me to [233].

Many thanks go to my colleagues in the Distributed Computing Group. First of all, I would like to thank Roland Flury for having been a super office mate—and Zurich marathon competitor! I am grateful for the generous help he offered on so many occasions, and I will never forget the *SOLA Duo* race. Special thanks go to Fabian Kuhn, who was my diploma thesis advisor; his marvelous way of tackling problems and his enthusiasm was certainly a motivation for me to continue my studies as a PhD student. I thank Thomas Moscibroda for the fruitful collaboration which broadened my scientific horizon, for having been a pleasant Discrete Event Systems co-assistant, and for his sense of humor in various situations. Thanks also go to my—*without loss of generosity*!—most productive peer-to-peer collaborator Thomas “T.Lo.” Locher for having been an arch-accomplice in stealing bits, for the *Happy Times* with time-shifted subtitles, and for making the publications he co-authored a pleasure to read. I thank Yvonne Anne Oswald for her interest in Al Capone, the families, Gaussian ski rental on trees—and *Suannai*, for having been a great traveling companion in Australia, China, Japan—and *MMP*, and for challenging the all-time science quiz record on the plane to Bangkok. I also thank Remo Meier for making *Pulsar* better than the SIGCOMM reviewers were prepared to and for inviting me to the “Kanonenrohr Piste”, Michael Kuhn for challenging discrete logarithm cryptography not only at the university of Berne and for the diligent use of the “rote Telefon”, Pascal von Rickenbach for sorting out the Highway to Hell problem with me and for shedding light on the mysteries of the *Piazza del Popolo*, Aaron Zollinger for introducing me to the YMCA in Berkeley—and to a popular web search company, Keno Albrecht for the computer support and for the time2climb in the *Gaswerk*, Nicholas Burri for having been the only Töggeli partner with whom I could improve my ranking, Olga Goussevskaia for telling me when to “eat!” the opponent’s queen at the TIK institute’s (chess) summer fight, and Johannes Schneider for organizing, among many others things, the Indian Ooty trip and the Swiss 3 a.m./4190 m.a.s.l. Strahlhorn expedition. I would also like to thank the following former and new DCG members, office mates, people from the institute, collaborators, visitors, etc.: Andi Wetzel, Beat Futterknecht, Bernard Mans, Carmine Ventre, Caterina Sposato, Christoph Lenzen, Jin Li, Lino Sevcik, Marco Cicolini, Mario Strasser, Martin May, Miriam Fritz, Mirjam Wattenhofer, Monica Fricker, Philipp Sommer, Rainer Baumann, Regina O'Dell, Reto Spöhel, Roland Mathis, Simon Heimlicher, Simon Schlachter, Thierry Dussuet, Thomas Steingruber, Uri Nadav, and Yves Weber.

A number of student projects have been conducted in the context of this dissertation. It is great that (so far) four students, Raphael Eidenbenz, Thomas Locher, Remo Meier, and Yvonne Anne Oswald, have decided to start a PhD in our group as well. Moreover, it was interesting to witness how Dominik Grolimund and Luzius Meisser's Kangoo/Wuala system evolved from a student project into a business plan. I wish the two of them an exciting future! Special thanks go to Joest Smit for his appetite for pancakes (not only on the Üetliberg!), to Patrick Moor who did a great job implementing BitThief, to Dominic Meier for analyzing the "family factor" in all situations of life (good luck for the next "Dreikönigskuchen competition"!), and to David Mysicka for reverse-engineering eMule and giving the WebSheriff no reasons to complain. I offer my thanks to: Adrian Moos, Bakshi Nikhil, Baptiste Prêtre, Christoph Renner, David Huber, David Mysicka, Dennis Rietmann, Dominic Meier, Dominik Egger, Dominik Grolimund, Dorian Kind, Fabian Hugelshofer, Georgios Katsikatsos, Henrique Säuberli, Jan Kostka, Jean-Luc Geering (thanks for the multilingual post!), Jelan Ong, Joest Smit, Jun Li, Lukas Füllemann, Luzius Meisser, Martin Kos, Matthias Hobi, Mirwais Tayebi, Patrick Moor, Philip Frey, Raphael Eidenbenz, Raphael Schmid, Rashid Waraich, Remo Meier, Samuel Willimann, Sandra Brockmann, Serkan Bozyigit, Stefan Sieber, Stefan Weber, Thibaut Britz, Thomas Locher, and Yvonne Anne Oswald.

Last but not least, I am grateful to my family (especially to my parents Irma and Severin, and to my *nonni* Alfred and Romana *la befana— "basta contare fino a cento!"*) who supported me during my entire education, to Markus Kern for the interest in my work and for providing me with local information on so many destinations—also outside the *Kremlin on the Charles*, and to Helen Kurukulasuriya for rendering stop-overs in Manhattan extremely healthy and for proof-reading this dissertation. Finally, I thank Christiane Baumann for inviting me to this momentous computer science "Schnuppervorlesung" roughly ten years ago.

Contents

Chapter 1

Introduction

1.1 The Peer-to-Peer Computing Paradigm

The *peer-to-peer* (p2p) computing paradigm has drawn much attention from both Internet users and research communities in the past few years, and a large fraction of today's Internet traffic is due to p2p communication. Principles of peer-to-peer computing also play a role in the discussions for the design of a future Internet. The paradigm has its roots in local area network (LAN) file sharing which became popular in the mid-1970s and many multiuser games have operated in a p2p model [64]. In contrast to traditional client/server architectures, each machine, or *peer*, in the p2p network acts both as a consumer as well as a *contributor* of resources. Therefore, a network's capacity increases with the number of participants. In addition to scalability, due to the decentralized and self-organizing nature, p2p systems are potentially robust and feature a high availability.

Peer-to-peer technology is used in a plethora of applications today, most prominently in the *Skype* Internet telephony application [220] and in file sharing tools such as *eMule* [86] or *BitTorrent* [50]. Also many companies have started investing into this technology. It is interesting how p2p applications have evolved over time. A fundamental problem in p2p computing is to discover and search the decentralized resources which often do not have permanent IP addresses. The first file sharing tool *Napster* [171] was based on servers. While the downloads were peer-to-peer, the lookup operation was still centralized: upon a file request, the server returned a list of remote computers storing this file. Early *unstructured* peer-to-peer architectures such as *Gnutella* [102] introduced distributed lookup operations. These systems have only a few constraints on the overlay topology and the mapping of data items to peers. However, albeit Gnutella's success, there have been concerns about its scalability from the beginning; indeed, when *Napster* was unplugged in 2001, Gnutella broke down soon afterwards due to the inrush of former Napster users. Currently there is a trend towards *structured* peer-to-peer systems which actively maintain desirable network properties such as low diameter and low peer degree by connecting peers in a smart manner. Both file lookup and download are fully decentralized. Most of the so-called *distributed hash table* (DHT) systems today, for instance *CAN* [198], *Chord* [226], *Pastry* [207], or *Tapestry* [248], are based on two influential seminal works, namely the paper by Plaxton, Rajaraman, and Richa on *locality-preserving data management* in distributed environments [193], and the paper by Karger, Lehman, Leighton, Levine, Lewin, and Panigrahi on *consistent hashing* [122]. Each peer in the DHT acts as a server for a subset of data items. Peers have overlay identifiers—these IDs are sometimes computed by applying a hash function on the peers' IP addresses—which define the logical connections between the peers, that is, each peer maintains a *routing table* which includes the addresses of peers located in different areas of the ID space. The data items (e.g., files) have *keys* which are typically chosen from the same space as the peer identifiers. A data item is stored on the peer whose ID is closest to the item's key. Thus, data items can be found by a targeted routing on the *overlay network* defined by the peers' routing tables. For a more detailed introduction to the foundations of p2p computing, the reader is referred to the corresponding literature [18, 116, 158, 224, 229].

1.2 Some Applications

Besides Internet telephony and file sharing, we currently witness the emergence of many other applications, for instance *live* or *on demand* media streaming (e.g., *Joost* [117], *PPLive* [194], or *Zattoo* [247]). In the following, some sample applications are presented which we have implemented ourselves.

Pulsar P2P Streaming System

Peer-to-peer technology can be an attractive solution for streaming content providers: as the downloading clients help to distribute the stream to other clients by contributing their upload bandwidth to the system, there is no need for an expensive infrastructure at the content distributor. The p2p paradigm has the potential to democratize the streaming world in that it enables everyone to broadcast her own media content—similarly to how the world wide web revolutionized the distribution of information more than a decade ago: nowadays, everyone can publish her thoughts on her own blog or website at virtually no cost.

Peer-to-peer live streaming faces several challenges which do not exist in classic file-sharing applications. For instance, real-time playback deadlines have to be met, and there are increased demands on robustness under joins and leaves of peers. Our *Pulsar* [151] (cf Figure 1.1) live streaming protocol applies a data distribution scheme which seeks to minimize both delays and redundant transmissions. Concretely, the protocol combines low-latency *push operations* (where data is forwarded actively to adjacent peers) along a structured overlay with the flexibility of *pull operations* (where missing

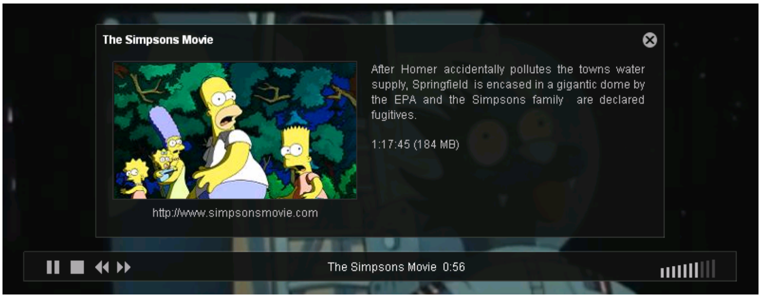

The Simpsons Movie
After Homer accidentally pollutes the towns water supply, Springfield is encased in a gigantic dome by the EPA and the Simpsons family are declared fugitives.
1:17:45 (184 MB)
http://www.simpsonsmovie.com
The Simpsons Movie 0:56

Figure 1.1: Screenshot of Pulsar applet.

data is requested explicitly by peers). In addition, the protocol incorporates mechanisms for locality awareness, incentive compatibility, and data integrity. Due to its small playback buffers, Pulsar can be emulated for up to 100,000 peers on a single desktop computer.

Pulsar is being extended for on-demand streaming. It is planned to use Pulsar to transmit video clips offered by the *tilllate.com* portal, and to broadcast DJ events.

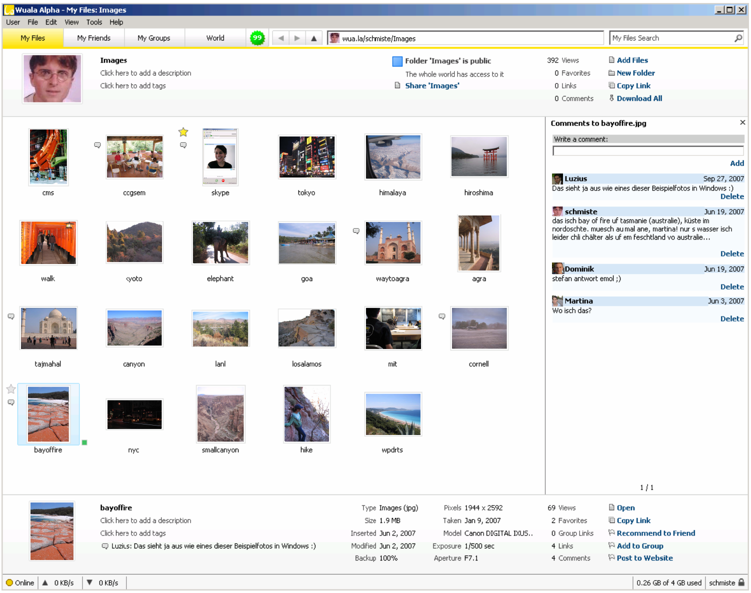
Wuala P2P Online Storage

Wuala [80] (cf Figure 1.2) is a global, distributed storage system that exploits idle disk space and bandwidth of participating computers to provide its users with a free and personal online storage. Files can be accessed from any computer. Besides storing private data on Wuala, a user can grant her friends access for certain files, or even share her holiday pictures with the entire community. Wuala's distributed hash table is based on principles of the *small world phenomenon* [234].

Wuala is not a pure p2p system as it relies on a server component for some operations, e.g., for certain security aspects. In addition, in order to prevent the unlikely case of data loss, the server maintains an encrypted copy of each file. However, much peer-to-peer technology is used to mitigate the load and cost at the server, and to boost download performance. Wuala also includes social networking features such as buddy lists. The system's fairness mechanism and its cryptographic file system have been described in [104, 105]. Wuala started as a student project and already has more than 12,000 users. Wuala is now marketed by the *Caleido* [56] company.

BitThief BitTorrent Client

BitThief [153] is a file sharing client for the BitTorrent system. In contrast to alternative clients, BitThief successfully avoids contributing resources to the p2p system. With this client, entire files can be obtained without up-



Images
Click here to add a description
Click here to add tags

Folder 'Images' is public
The whole world has access to it
Share 'Images'

392 Views
0 Favorites
0 Links
0 Comments

Comments to bayoffire.jpg
Write a comment:

Luzius Sep 27, 2007
Das sieht ja aus wie eines dieser Beispielfotos in Windows :)

schmiste Jun 19, 2007
das isch bay of fire uf tasmanie (australie), küste im nordoschte. muesch au mal ane, martina! nur s wasser isch leider chli chälter als uf em feschtland vo australie...

Dominik Jun 19, 2007
stefan antwort emol ;)

Martina Jun 3, 2007
Wo isch das?

bayoffire
Click here to add a description
Click here to add tags
Luzius: Das sieht ja aus wie eines dieser Beispielfotos in Windows :)

Type Images (jpg)
Size 1.9 MB
Inserted Jun 2, 2007
Modified Jun 2, 2007
Backup 100%

Pixels 1944 x 2592
Taken Jan 9, 2007
Model Canon DIGITAL IXUS..
Exposure 1/500 sec
Aperture F7.1

69 Views
2 Favorites
0 Group Links
4 Links
4 Comments

Online 0 KB/s 0 KB/s 0.26 GB of 4 GB used schmiste

Figure 1.2: Screenshot of Wuala.

loading any useful data. This thesis argues that the design of mechanisms to enforce collaboration among peers is an important research challenge today. BitThief is a proof-of-concept software that demonstrates that today's BitTorrent system does not fend off non-cooperative peers effectively. BitThief is described in detail in Chapter 8.

Distributed Compilation at CERN

The *European Organization for Nuclear Research* (CERN) builds large amounts of software for its experiments at the *Large Hadron Collider* (LHC), and it can be useful to distribute the compilation jobs in order to shorten project build times. This is possible by taking advantage of idle desktop machines available as a virtual build cluster [23]. Due to the clear distinction between the compilation server and the compiling desktop clients, the proposed solution is not a p2p system in the classic sense. However, there are several commonalities; for instance, the participating clients are heterogeneous and can join and leave frequently, e.g., whenever the screensaver is switched on or off, respectively.

Peer-to-Peer XPilot

Multi-player games can make use of peer-to-peer technology as well, as traditionally, powerful and expensive game servers are needed to manage the large number of players. However, designing p2p multi-player games is challenging: it is crucial that the peers have a consistent and synchronous view of the game's state, and that the latency in the game be minimized. Additionally, the game must be cheater-resistant. These properties are often difficult to achieve in a purely distributed environment. We have gained first experiences by porting *XPilot* to the post-client/server world [115].

Crypto-Cracker for BOINC

The *grid computing* paradigm has many similarities with the peer-to-peer paradigm. Grids aggregate the computational power of thousands of machines to solve large mathematical problems. The computational tasks or units are typically distributed to the clients by a central server. Similarly to the peers in p2p networks, the grid's clients are often under the control of individual users and may therefore be dynamic and behave selfishly. In fact, such strategic behavior has already occurred in computational peer-to-peer settings; for instance, users of the SETI project of UC Berkeley's Space Sciences Laboratory have modified their software to make it appear as if they were doing more work [216].

We have implemented a client for the *Berkeley Open Infrastructure for Network Computing* (BOINC) framework [15]. Our client allows us to attack hard crypto-puzzles such as the discrete logarithm problem in large *elliptic curves* [61, 62]. We have extended the BOINC server software such that the participants can verify each other's results [137]. This checking is needed to unmask cheaters returning wrong results, for instance, to earn credit points without actually contributing any CPU cycles.

1.3 Algorithmic Challenges

Peer-to-peer computing today still faces many challenges. In contrast to other multi-computer architectures which are typically quite static and where faulty machines may sometimes even be repaired manually without much loss of availability, peer-to-peer networks are typically highly dynamic. While traditional systems were based on fixed infrastructures and were under a single administrative domain (e.g., owned and maintained by a single company or corporation), the participating machines in p2p networks are under the control of individual (and to some extent: anonymous) users who can join and leave at any time and concurrently. The peer-to-peer computing paradigm heralded a new era in the sense that faults in and extensions/additions to the network were no longer considered a cumbersome problem but a main design principle, and a means to gather idling resources on the "edge of the Internet". Therefore, in p2p networks, many system components have to incorporate mechanisms to cope with (or even exploit) these dynamic changes of the available resources.

Peers in a p2p network do not only join and leave arbitrarily, but they can also crash. In p2p parlor, these frequent membership changes are called *churn*. As an example, consider our live streaming tool Pulsar. Shortly before the start of a soccer event, many users will join the system. During the transmission, there is less churn, but some users still join and some leave, e.g., because their favorite team plays poorly, or because their computer loses access to the Internet. At the end of the game, it is likely that many peers disconnect from the network in a short time period.

We believe that ensuring a seamless system operation despite churn is a key challenge in p2p computing. As churn is a new phenomenon in computer science, not many results from classic CS literature are directly applicable, making p2p research a field of its own merit.

A second implication of the autonomy of the machines in p2p networks is that the system consists of different *stakeholders*. Users can have various reasons for joining the network. For instance, an (anonymous) user may not voluntarily contribute her bandwidth, disk space, or CPU cycles to the system, but prefer to free ride. Hence, the fact that the peers are under the control of individual users adds a socio-economic aspect to peer-to-peer computing. As the p2p paradigm relies on the contributions of the participating machines, effective incentive mechanisms have to be designed which foster cooperation and punish free riders.

We regard cooperation as a crucial challenge in peer-to-peer computing. Economic models have only recently gained the attention of the CS community. However, due to the topic's importance—not only for p2p computing but for the Internet in general—much research has been conducted over the last years. This research could build upon the foundations developed by mathematicians and economists earlier in the century. In this thesis, we seek to apply these techniques to peer-to-peer systems. Our tools help to identify weaknesses in a given system which could potentially be exploited by self-interested users; armed with this knowledge, appropriate countermeasures can be designed.

The cooperation challenge goes beyond the free rider problem, and there are many other forms of non-cooperation. For instance, some users may try to harm the system, independently of their cost. In the following, we will call such participants "malicious", and we will regard robustness against malicious attacks as desirable.[1] Coping with malicious users is in some sense simple as pure p2p systems are fully decentralized and do not have any bottlenecks or single points of failure. On the other hand, however, unlike in the real world or society, a single malicious user can increase her influence in the network by creating a large number of (virtual) identities or by hijacking and taking control over other machines in the system. For instance, a participant could try to censor contents in a p2p network by actively removing all network pointers to it.

This thesis focuses on the two arguably most intriguing challenges in p2p computing: *dynamics* and *cooperation*. Of course, there are additional in-

[1] Of course, we are aware that p2p technology can also be abused, e.g., to illegally distribute copyrighted material or to organize botnets. However, in the following, we will assume a system designer's perspective.

teresting and novel aspects. For instance, peers are typically geographically distributed all over the planet and latencies must not be neglected. Another crucial implication of the p2p paradigm is that—unlike in traditional systems—participating machines can be very *heterogeneous* (e.g., [49, 162] or [213]): peers can be located in different countries, have different hardware and different Internet connections, run different operating system, and so on. A successful p2p system must be able to deal with these heterogeneous resources and exploit them optimally. Note that, in general, it is difficult to define what "dealing with heterogenous resources" means. It could mean that peers which contribute more should also get a better service from the system. Alternatively, a system designer could aim at providing the same service to all peers by using extra resources of "strong" peers to compensate for the lack of resources of "weak" peers. The goal of dealing effectively with heterogeneous participants can also conflict with the cooperation goal: how can we distinguish a weak peer, which is not able to contribute more resources, from a selfish peer which seeks to free ride? For instance, in Pulsar, we have decided not to distinguish these two kinds of peers, but to exclude any peer which seems to be incapable of uploading at the rate of the currently streamed media. Apart from attacks by malicious participants or *insiders*, a p2p network can be threatened by entities outside the network. Such attacks also constitute an important research challenge. Another challenge in p2p computing is the interplay between p2p users and ISPs [7]. On the one hand, p2p systems have resulted in an increase in revenue for ISPs, as users have upgraded their Internet access to broadband. On the other hand, p2p applications pose a traffic engineering challenge at the ISPs, as the peers' overlay network is largely independent of the Internet topology, and as without countermeasures, traffic often crosses network boundaries multiple times. Since most Internet bottlenecks are assumed to be either in the access network or on links between ISPs, this is problematic [10]. *User anonymity* is also an interesting problem in p2p computing (cf also the *Freenet* system [65]): e.g., a system designer could try to ensure freedom of speech by protecting the identities of bloggers living under an undemocratic regime. Many challenges in the area of distributed computing are also present in p2p systems. Besides well-known impossibility results such as achieving *consensus* among dynamic players (e.g., on the current state of a p2p game such as XPilot), many cryptographic protocols are more complex in the absence of any trusted and centralized party. Last but not least, p2p computing faces some "soft" challenges. Many companies jumped on the "free bandwidth with BitTorrent" bandwagon[2] passing on costs to Internet service providers and eventually consumers, which (economically) threatens the paradigm. Additionally, today's p2p systems have to compete with inexpensive server solution (e.g., *Amazon's Simple Storage Service S3*). Despite the importance of all these challenges, we will not further investigate them in this thesis but refer the reader to the corresponding literature.

[2]BitTorrent Inc.'s partners include 20th Century Fox, MTV, Paramount, Warner Brothers, among many others.

1.4 Thesis Overview

The thesis is organized in two parts. Part I addresses the challenge of dynamics in p2p systems. After a short introduction in Chapter 2, Chapter 3 presents algorithms which provably maintain desirable properties of peer-to-peer networks despite ongoing churn. We assume a worst-case perspective and consider an adversary who can add and remove peers at arbitrary places in the network. Unlike in traditional algorithmic research, joins and leaves occurring in a worst-case manner in p2p systems are hardly considered in the literature so far. Although membership changes in reality may be less biased and follow probabilistic distributions, our conservative approach gives stronger guarantees; moreover, our model also includes scenarios where the peers' joins and leaves are orchestrated by an attacker or virus. Chapter 3 describes algorithms to maintain a hypercubic p2p network. It is shown that although our system may never be fully repaired in such a scenario, it is always fully functional. Concretely, we prove that despite an adversary who can add and remove $\Theta(\log n)$ peers per round, both the peer degree (i.e., the number of neighbors of a peer in the overlay network) and the network diameter are bounded by $O(\log n)$, where n is the total number of peers in the system. This is asymptotically optimal in the sense that no p2p system can have a lower peer degree in the presence of such an adversary. The algorithms can be generalized for alternative network topologies, and Chapter 3 also sketches how to adapt the algorithms for pancake graphs. The resulting p2p system maintains peer degree and network diameter $O(\log n / \log \log n)$ against an adversary who performs $\Theta(\log n / \log \log n)$ membership changes per round. This is also asymptotically optimal.

Chapter 4 then looks at a different form of dynamics, namely dynamic changes of the available bandwidth between two peers. A transfer protocol is described which seeks to maximize throughput in this environment. Again, we assume a conservative perspective and consider a scenario where the available bandwidth is changed by an adversary. We also assume that the transfer protocol does not know the available bandwidth in the future and has to decide on the transmission rate *online*. We are hence in the realm of online algorithms and competitive analysis. The contributions of Chapter 4 are twofold: first, a new and simple analysis is presented for an existing adversary model. Second, a new model for the dynamics is introduced. It is based on network calculus concepts and allows for bursty changes of the available bandwidth. We believe that this model can be instructional in many other disciplines also dealing with dynamics. After reviewing related literature on p2p dynamics in Chapter 5, we conclude Part I in Chapter 6.

Part II of this thesis studies the challenge of cooperation in p2p computing. After a short introduction (Chapter 7), we motivate the topic's importance with a case study of BitTorrent in Chapter 8. BitTorrent is widely believed to incorporate effective means to fend off selfish peers. Contrary to such belief, we show that BitTorrent's tit-for-tat incentive mechanism can be cheated and that it is possible to free ride. Our proof-of-concept client *Bit-Thief* achieves good download rates without uploading any real data, even in the absence of seeder peers. It is also shown in Chapter 8 that sharing communities can be exploited. We believe that the distribution of such a client can be harmful for BitTorrent's performance.

In order to study the impact of selfish behavior in p2p computing, we give a game-theoretic analysis in Chapter 9. We assume an unstructured p2p network and analyze the effects if peers selfishly select to which other peers they connect. The analysis reveals that the so-called *Price of Anarchy* is $\Theta(\min(\alpha, n))$, where α is a parameter that captures the tradeoff between lookup performance (low stretches) and the cost of neighbor maintenance, and n is the number of peers in the system. In other words, the efficiency of the resulting p2p topologies suffers for large networks if α is large. Chapter 9 also shows that selfishness can cause the system to become instable, and that deciding whether a given p2p system has a so-called pure Nash equilibrium is **NP**-hard, i.e., it is computationally at least as hard a solving a special type of a Boolean satisfiability problem. In Chapter 10, a mathematical framework is introduced which extends the concepts used in game theory to take malicious behavior into consideration. Our framework allows us to quantify the effect of malicious peers in a network. We give a sample analysis for a virus inoculation game. It turns out that in a scenario where peers are not aware of the presence of malicious players, the system performance deteriorates. Intriguingly, however, we observe that in a non-oblivious model where peers are risk-averse, adding malicious players may be beneficial to the system compared to purely selfish environments. Our tools of Chapter 10 allow to quantify the so-called *Fear Factor*, and an upper bound for this factor is derived. As a short excursion, Chapter 11 demonstrates how the framework of Chapter 10 can be used also in other contexts, namely in *social networks*. In these networks, nodes (individuals or organizations) are tied by a specific type of interdependency, e.g., friendship, economic trade, or disease transmission. We again study the virus inoculation game, but now in an environment where players *care* about the well-being of their direct social contacts, e.g., their friends stored in Skype's contact lists. We introduce the notion of a *Windfall of Friendship* and show that a system can never suffer if users act socially. However, interestingly, the social costs do not decrease monotonically in the extent to which players care about each other.

In Chapter 12, our theoretic insights into the effects of malicious behavior are complemented by a study of possible attacks in today's *Kad network*—currently the most widely deployed p2p system based on a DHT, with more than a million simultaneous users. Several protocol exploits are described which prevent peers from accessing certain files, e.g., by hijacking file requests. Alternatively, we show that it is possible to overwhelm publishing peers with bogus information such that pointers to the original files can no longer be accessed. It is even possible in Kad to eclipse certain peers, i.e., to fill up their routing tables with information about malicious peers. We will also briefly discuss how our attacks can be used to harm machines *outside* the Kad network, e.g. web servers, by tricking the peers into performing a Kad-steered distributed denial of service (DDoS) attack. Related work on the threat of non-cooperation is discussed in Chapter 13. Finally, Part II is concluded in Chapter 14.

Part I

Dynamics

Chapter 2

Introduction

Distributed systems appear in many different forms and are well-known also outside the CS world. For instance, the human brain can be regarded as a distributed system, or the actors on a given economic market, or even society in general. A common property of such networks is that they are often complex already in the static case. However, in reality, these systems are typically highly dynamic in various dimensions: connections between cells in the brain change over time, new actors enter a market while others leave, bubbles emerge in a financial sector and subsequently implode, shifts in society can lead to the election of a new party or the adoption of a new form of government, and so on. This renders the analysis of a distributed system more difficult, and currently—and probably still in the near future—much research is conducted to gain deeper insights into these dynamic aspects. For instance, in a recent study by Leskovec, Kleinberg, and Faloutsos [143] of different types of social networks, such as email graphs, patents citation graphs, or autonomous systems graph, it has been discovered that most of these networks *intensify* over time, with the number of edges growing superlinearly in the number of nodes. Moreover, the average distance between nodes shrinks over time.

Understanding and coping with dynamics is particularly important in peer-to-peer computing as p2p networks are often large and peers can connect for short time periods only. For example, in a network consisting of one million peers and where each peer remains in the system for one hour, we expect roughly 300 membership changes per second. Any p2p system therefore needs a maintenance protocol which continuously repairs the network. Due to the churn, the system is almost never in its "ideal state", and it has to be ensured that peer-to-peer services like lookup operations are correct even on imperfect topologies. Besides the membership dynamics, other properties of a p2p network change over time as well—for instance the bandwidth available between two peers.

So far, only few fundamental principles have been found which help to render p2p networks robust to churn. One such principle is the use of a meaningful amount of *redundancy*. Only a data item which is replicated on other peers can survive the departure of the hosting peer. Moreover, in contrast to classic parallel computers which were often based on constant-degree topologies such as *butterfly networks* [141], there is a trend towards logarithmic-degree topologies which are more fault-tolerant in the sense that it is harder to separate a set of peers from the rest of the network. Another important principle is *locality* [187]: a peer should decide on its future actions solely depending on the states of the peers in its *vicinity*. Only such *locality-sensitive* distributed algorithms where communication is restricted to close neighborhoods are fast enough to allow peers to react quickly to changing environments.

Part I of this thesis studies how to cope with dynamics in p2p systems. In Chapter 3, we will develop algorithms which provably maintain desirable properties of peer-to-peer networks despite ongoing churn. These algorithms are based on a so-called token distribution algorithm which uniformly distributes peers on the identifier space, and on an aggregation algorithm which estimates the total number of peers in the system. It is first shown how to apply our techniques to a hypercubic p2p network with peer degree and network diameter $O(\log n)$. Chapter 3 also describes how to adapt the algorithms to maintain a pancake graph with peer degree and network diameter $O(\log n / \log\log n)$. Chapter 4 then considers an optimization problem arising in the context of p2p file downloads and devises transfer protocols which selfishly seek to maximize throughput despite dynamic changes of the available bandwidth. This chapter also introduces a novel adversarial model for dynamic systems exhibiting changes with bursts.

Chapter 3

Worst-Case Churn

Due to the rise of peer-to-peer systems, sensor networks, and mobile ad hoc networks, logical networks, or *overlay networks*, are becoming more and more wide spread. A major complication in these networks is that they can be highly dynamic, which requires fast and robust recovery mechanisms.

Today, the analysis of fault tolerance of p2p systems usually only covers random faults of some kind. Contrary to traditional algorithmic research, faults as well as joins and leaves occurring in a worst-case manner in p2p systems are hardly considered. Moreover, most fault tolerance analyses are static in the sense that only a functionally bounded number of random peers can be crashed. After removing a few peers the system is given sufficient time to recover again. The much more realistic dynamic case where faults steadily occur has not found much attention.

This chapter studies dynamic worst-case joins and leaves. We think of an *adversary*[1] who cannot be fooled by any kind of randomness. The adversary can choose which peers to crash and how peers join. Moreover, the adversary does not have to wait until the system is recovered before crashing the next batch of peers. Instead, the adversary can constantly crash peers while the system is trying to stay alive. Indeed, the (structured) peer-to-peer system presented in this chapter is never fully repaired but always fully functional. Our system is resilient against an adversary who continuously attacks the "weakest part" of the system. Such an adversary could for example insert a crawler into the p2p system, learn the topology, and then repeatedly crash selected peers, in an attempt to partition the network. Our system counters such an adversary by continuously moving the remaining or newly joining peers towards the sparse areas.

Of course, we cannot allow our adversary to have unbounded capabilities. In any constant time interval, the adversary can at most add and/or remove $O(\log n)$ peers, n being the total number of peers currently in the system. This model covers an adversary who repeatedly takes down machines by a

[1]Note that henceforth, we will use the term "adversarial" to denote worst-case behavior.

distributed denial of service attack, however only a logarithmic number of machines at each point in time. Our maintenance algorithm relies on messages being delivered timely, in at most constant time between any pair of operational peers. In distributed computing, such a system is called *synchronous*. Note that if nodes are synchronized locally, our algorithm also runs in an asynchronous environment. In this case, the propagation delay of the slowest message defines the notion of time which is needed for the adversary model. In our synchronous message passing model, each peer can send a message to all its neighbors in each round. In the following, we will refer to an adversary who can perform J arbitrary joins and and L arbitrary leaves (crashes) in each interval of λ rounds by $\mathcal{A}(J, L, \lambda)$.

The basic structure of our p2p system is a (simulated) *hypercube*: each peer is part of a distinct hypercube node; each hypercube node consists of $\Theta(\log n)$ peers. Peers have connections to other peers of their hypercube node and to peers of the neighboring hypercube nodes. After a number of joins and leaves, some peers may have to change to another hypercube node such that up to constant factors, all hypercube nodes have the same cardinality at all times. If the total number of peers grows or shrinks above or below a certain threshold, the dimension of the hypercube is increased or decreased by one, respectively.

The balancing of peers among the nodes can be seen as a *dynamic token distribution problem* on the hypercube. Each node of a graph (hypercube) has a certain number of tokens, and the goal is to distribute the tokens along the edges of the graph such that all nodes end up with the same or almost the same number of tokens. While tokens are moved around, an adversary constantly inserts and deletes tokens.

Our p2p system builds on two basic components: i) an algorithm which performs the described dynamic token distribution and ii) an information aggregation algorithm which is used to estimate the number of peers in the system and to adapt the hypercube's dimension accordingly.

Based on the described structure, we get a system which tolerates $O(\log n)$ worst-case joins and/or crashes per constant time interval. As in other p2p systems, peers have $O(\log n)$ neighbors, implying that the achieved fault-tolerance is asymptotically optimal. The usual operations (e.g. search) take time $O(\log n)$.

3.1 Algorithmic Components

Our maintenance algorithm for the simulated hypercube system ensures that each node always contains at least one peer which stores the node's data. Further, it adapts the hypercube dimension to the total number of peers in the system. This is achieved by a dynamic token distribution algorithm on the hypercube and an information aggregation scheme which allows the nodes to simultaneously change the dimension of the hypercube. These two components are described in turn.

3.1.1 Dynamic Token Distribution

The problem of distributing peers uniformly throughout a hypercube is a special instance of a *token distribution problem*, first introduced by Peleg and Upfal [188]. The problem has its origins in the area of *load balancing*, where the workload is modeled by a number of *tokens* or jobs of unit size; the main objective is to distribute the total load equally among the processors. Such load balancing problems arise in a number of parallel and distributed applications including *job scheduling* in operating systems, *packet routing*, large-scale *differential equations* and parallel *finite element methods*. More applications can be found in [215].

Formally, the goal of a token distribution algorithm is to minimize the maximum difference of tokens at any two nodes, denoted by the *discrepancy* ϕ. This problem has been studied intensively; however, most of the research is about the *static variant* of the problem, where given an arbitrary initial token distribution, the goal is to redistribute these tokens uniformly. In the *dynamic variant* on the other hand, the load is dynamic, that is, tokens can arrive and depart *during* the execution of the token distribution algorithm. In our case, peers may join and leave the hypercube at arbitrary times, so the emphasis lies on the dynamic token distribution problem on a d-dimensional hypercube topology.

We study two types of the token distribution problem: in the *fractional token distribution*, tokens are arbitrarily divisible, whereas in the *integer token distribution*, tokens can only move as a whole. In our case, tokens represent peers and are inherently integer. However, it turns out that the study of the fractional model is useful for the analysis of the integer model.

Our token distribution algorithm is based on the *dimension exchange method* [70, 192]. Basically, the algorithm cycles continuously over the d dimensions of the hypercube. In step s, where $i = s \bmod d$, every node $u = \beta_0 ... \beta_i ... \beta_{d-1}$ having a tokens balances its tokens with its adjacent node in dimension i, $v = \beta_0 ... \overline{\beta_i} ... \beta_{d-1}$, having b tokens, such that both nodes end up with $\frac{a+b}{2}$ tokens in the fractional token distribution. On the other hand, if the tokens are integer, one node is assigned $\lceil \frac{a+b}{2} \rceil$ tokens and the other one gets $\lfloor \frac{a+b}{2} \rfloor$ tokens.

It has been pointed out in [70] that the described algorithm yields a perfect discrepancy $\phi = 0$ after d steps for the static fractional token distribution. In [192], it has been shown that in the worst case, $\phi = d$ after d steps in the static integer token distribution.

In the following, the dynamic integer token distribution problem is studied, where a "token adversary" $\mathcal{A}(J, L, 1)$ adds at most J and removes at most L tokens at the beginning of each step. In particular, we will show that if the initial distribution is perfect, i.e., $\phi = 0$, our algorithm maintains the invariant $\phi \leq 2J + 2L + d$ at every moment of time.

For the dynamic fractional token distribution, the tokens inserted and deleted at different times can be treated independently and can be superposed. Therefore, the following lemma holds.

Lemma 3.1. *For the dynamic fractional token distribution, the number of tokens at a node depends only on the token insertions and deletions of the last d steps and on the total number of tokens in the system.*

Proof. Assume that a total amount of T tokens are distributed in two different ways on the d-dimensional hypercube. According to [70], in the absence of the adversary, each node has exactly $T/2^d$ tokens after d steps. On the other hand, the token insertions and removals of the adversary that happen in-between can be treated as an independent superposition, as the corresponding operations are all linear. $\square$

We can now bound the discrepancy of the integer token distribution algorithm by comparing it with the fractional problem.

Lemma 3.2. *Let v be a node of the hypercube. Let $\tau_v(t)$ and $\tau_{v,f}(t)$ denote the number of tokens at v for the integer and the fractional token distribution algorithm at time t, respectively. We have $\forall t : |\tau_v(t) - \tau_{v,f}(t)| \leq d/2$.*

Proof. For $t = 0$, we have $\tau_v(t) = \tau_{v,f}(t)$. For symmetry reasons, it is sufficient to show the upper bound $\tau_v(t) \leq \tau_{v,f}(t) + d/2$. We first prove by induction that $\tau_v(t) \leq \tau_{v,f}(t) + t/2$ at time t.

For the induction step, we consider two neighbors u and v which exchange tokens. We have

$$\begin{aligned}
\tau_v(t+1) &\leq \left\lceil \frac{\tau_v(t) + \tau_u(t)}{2} \right\rceil \\
&\leq \left\lceil \frac{\lfloor \tau_{v,f}(t) + \frac{t}{2} \rfloor + \lfloor \tau_{u,f}(t) + \frac{t}{2} \rfloor}{2} \right\rceil \\
&\leq \frac{\lfloor \tau_{v,f}(t) + \frac{t}{2} \rfloor + \lfloor \tau_{u,f}(t) + \frac{t}{2} \rfloor}{2} + \frac{1}{2} \\
&\leq \tau_{v,f}(t+1) + \frac{t+1}{2}.
\end{aligned}$$

The second inequality follows from the induction hypothesis and the fact that $\tau_v(t)$ and $\tau_u(t)$ are integers. Note that adding or removing tokens has no influence on the difference between τ_v and $\tau_{v,f}$ because it modifies τ_v and $\tau_{v,f}$ in the same way.

So far, we have seen that the number of integer tokens can deviate from the number of fractional tokens by at most $d/2$ after the first d steps. In order to show that this holds for all times t, we consider a fractional token distribution problem $\hat{\tau}_{v,f}$ for which $\hat{\tau}_{v,f}(t-d) = \tau_v(t-d)$. Using the above argument, we have $\tau_v(t) \leq \hat{\tau}_{v,f}(t) + d/2$ and by Lemma 3.1, we get $\hat{\tau}_{v,f}(t) = \tau_{v,f}(t)$. This concludes the proof. $\square$

Lemma 3.3. *In the presence of an adversary $\mathcal{A}(J, L, 1)$, for the integer discrepancy, it always holds that $\phi \leq 2J + 2L + d$.*

Proof. We show that the *fractional* discrepancy ϕ_f is bounded by $2J + 2L$. Since Lemma 3.2 implies that for the integer discrepancy ϕ_i it holds that $\phi_i - \phi_f \leq d$, the claim follows. Let $J_t \leq J$ and $L_t \leq L$ be the insertions and deletions that happen at the beginning of step t. First, we consider the case of joins only, i.e., $L_t = 0$. Assume that all J_t tokens are inserted at node $v = \beta_0 ... \beta_i ... \beta_{d-1}$ where $i = t \bmod d$. In the upcoming paragraph, all indices are implicitly modulo d. In step t, according to the token distribution algorithm, v keeps $J_t/2$ tokens and sends $J_t/2$ to node $u = \beta_0 ... \overline{\beta_i} ... \beta_{d-1}$. In step $t+1$, $J_t/4$ tokens are sent to nodes $\beta_0 ... \beta_i \overline{\beta_{i+1}} ... \beta_{d-1}$ and $\beta_0 ... \overline{\beta_i \beta_{i+1}} ... \beta_{d-1}$, and so on. Thus, after step $t+d-1$, every node in the d-dimensional hypercube has the same share of $J_t/2^d$ tokens from that insertion. We conclude that a node can have at most all insertions of this step, half of the insertions of the last step, a quarter of all insertions two steps ago, and so on:

$$\underbrace{J_t + \frac{J_{t-1}}{2} + \frac{J_{t-2}}{4} + ... + \frac{J_{t-(d-1)}}{2^{d-1}}}_{< 2J} + \underbrace{\frac{J_{t-d}}{2^d} + \frac{J_{t-(d+1)}}{2^d} + \frac{J_{t-(d+2)}}{2^d} + ...}_{\text{shared by all nodes}}$$

Since $J_{t-i} \leq J$ for $i = 0, 1, 2, \ldots$, we have $\phi_f \leq 2J$. For the case of only token deletions, the same argument can be applied, yielding a discrepancy of at most $2L$. Finally, if there are both insertions and deletions which do not cancel out each other, we have $\phi_f \leq 2J + 2L$. □

3.1.2 Remark on Random Token Distribution

Observe that if the decision to which node to assign $\lceil \frac{a+b}{2} \rceil$ tokens and to which node to assign $\lfloor \frac{a+b}{2} \rfloor$ tokens is made uniformly at random with probability .5, the final (static) integer discrepancy of our algorithm is even constant in expectation.

Theorem 3.4. *Let X be the random variable for the final discrepancy in a d-dimensional hypercube. It holds that $\mathbb{E}[X] < 3$.*

Proof. In our randomized rounding scheme, the rounding direction of each edge it determined by a perfect coin flip. Let X_i be the random variable denoting the number of incoming edges of a $(d-1-i)$-dimensional sub-cube. Since there are 2^{d-1-i} edges connecting two $(d-1-i)$-dimensional sub-cubes H_0 and H_1, X_i is *binomially* distributed: $X_i \sim Bin(2^{d-1-i}, 1/2)$. If rounding happens on every edge, the sub-cubes H_0 and H_1 differ by δ_i tokens after balancing, where $\delta_i = 2 \cdot |X_i - \mathbb{E}[X_i]|$, and the random variables δ_i are mutually independent.

Assume that in the final distribution, the maximum node v has a tokens. We show that the *average* number of tokens in the system is at least $a - \sum_{i=0}^{d-1} 2^i \mathbb{E}[\delta_i]/2^d$, by counting the average number of tokens in the biggest i-dimensional sub-cubes which contain v for $i \in [0, d]$. Obviously, the 0-dimensional sub-cube consists only of v and has a tokens in total. In the next step, this sub-cube is combined with another 0-dimensional sub-cube having $a - \delta_{d-1}$ tokens. The resulting 2-dimensional hypercube having $2a - \delta_{d-1}$ tokens is combined with a hypercube having δ_{d-2} tokens less, hence there are $4a - 2\delta_{d-1} - \delta_{d-2}$ tokens in total, and so forth. After d steps, we have $a2^d - \sum_{i=0}^{d-1} 2^i \delta_i$ tokens in the whole d-dimensional hypercube, $a - \sum_{i=0}^{d-1} 2^i \delta_i / 2^d$ on average.

It remains to calculate $\mathbb{E}[\delta_i]$, which is twice the *mean deviation* of the binomial distributed random variable X_i. In the following, we will first give a proof using a *Chernoff* argument, and then derive a better bound by a *Stirling* approximation.

We will first derive this upper bound on the mean of δ_i:

$$\mathbb{E}[\delta_i] = 2 \cdot \frac{1}{2^{2^{d-1-i}}} \cdot \sum_{j=0}^{2^{d-1-i}} \binom{2^{d-1-i}}{j} \left| j - \frac{2^{d-1-i}}{2} \right| \overset{(1)}{\leq} 2\sqrt{\pi 2^{d-1-i}}$$

Inequality (1) is a Chernoff approximation. In fact, we need the following two facts, Fact 3.1 and Fact 3.2, plus Lemma 3.5.

Fact 3.1 (Chernoff Lower Tail). *Let $X_1, ..., X_N$ be independent Bernoulli variables with $\mathbb{P}[X_i = 1] = p_i$. Let $X = \sum_i X_i$ denote the sum of the X_i and let $\mu = \mathbb{E}[X] = \sum_i p_i$ be the expected value for X. For $\epsilon \in (0, 1]$,*

$$\mathbb{P}[X < (1-\epsilon)\mu] < \left(\frac{e^{-\epsilon}}{(1-\epsilon)^{(1-\epsilon)}} \right) < e^{-\mu\epsilon^2/2}.$$

Fact 3.2.

$$\int_0^\infty e^{-x^2} dx = \frac{\sqrt{\pi}}{2}.$$

Lemma 3.5. *Let $X \sim Bin(n, 1/2)$ be binomially distributed with parameters n and $p = 1/2$. The expectation of the deviation from the mean $n/2$ is upper bounded by*

$$\mathbb{E}[|X - n/2|] \leq \sqrt{\pi n}.$$

Proof. Let p_δ denote the probability that the deviation from the mean is at least δ, that is, $p_\delta = \mathbb{P}[|X - n/2|] \geq \delta$. By symmetry, we have $p_\delta = 2 \cdot \mathbb{P}[X \leq n/2 - \delta]$. For the expected deviation of the mean, we have

$$\mathbb{E}[|X - n/2|] = \sum_{\delta=1}^{n/2} \delta \cdot \mathbb{P}[|X - n/2| = \delta] = \sum_{\delta=1}^{n/2} p_\delta. \tag{3.1}$$

We can bound p_δ using Chernoff:

$$p_\delta = 2 \cdot \mathbb{P}[X \leq n/2 - \delta] \leq 2e^{-\delta^2/n}. \tag{3.2}$$

Combining (3.1) and (3.2), we can bound the mean deviation by

$$\begin{aligned}\mathbb{E}[|X-n/2|] &= \sum_{\delta=1}^{n/2} p_\delta \le 2\cdot\sum_{\delta=1}^{n/2} e^{-\delta^2/n}\\ &< 2\cdot\sum_{\delta=1}^{\infty} e^{-\delta^2/n}\\ &< 2\cdot\int_{\delta=1}^{\infty} e^{-\delta^2/n} d\delta\\ &= \sqrt{\pi n}.\end{aligned}$$

The integral after the last inequality can be calculated using Fact 3.2 and the substitution $t=\delta\sqrt{n}$ and $d\delta=\sqrt{n}dt$. This concludes the proof. □

Remarks: *There are two minor details which are neglected for readability of the above proof. First, although the Chernoff inequality gives an upper bound only for $\mathbb{P}[X<(1-\epsilon)\mu]$, we use it for $\mathbb{P}[X\le(1-\epsilon)\mu]$. Second, the first equation in the proof holds for even n. For odd n, the deviation from the mean is not integral. Both issues can easily be solved.*

Given the bound on $\mathbb{E}[\delta_i]$, we can compute the token sum

$$\begin{aligned}\sum_{i=0}^{d-1} 2^i\mathbb{E}[\delta_i] &= 2\sqrt{\pi}\sum_{i=0}^{d-1} 2^i 2^{\frac{d-1-i}{2}} = 2\sqrt{\pi}2^{\frac{d-1}{2}}\sum_{i=0}^{d-1} 2^{\frac{i}{2}}\\ &= 2\sqrt{\pi}2^{\frac{d-1}{2}}\sum_{i=0}^{d-1}(\sqrt{2})^i = 2\sqrt{\pi}2^{\frac{d-1}{2}}\frac{(\sqrt{2})^d-1}{\sqrt{2}-1}\\ &\le \frac{\sqrt{\pi}}{\sqrt{2}-1}2^d\end{aligned}$$

Thus, having a node with a tokens, the average number of tokens is at least $a-\frac{\sqrt{\pi}}{\sqrt{2}-1}$, and by symmetry, the expected final discrepancy is twice as much.

We obtain an even better bound by using *Stirling's approximation* rather than applying Chernoff. In the following, we show that we overestimated by a factor of at least π, and therefore the total discrepancy is $2\cdot\frac{1}{\pi}\cdot\frac{\sqrt{\pi}}{\sqrt{2}-1}\doteq 2.73$.

The mean deviation MD of the symmetrical binomial distribution is given by:

$$\mathrm{MD} = 2^{-n}\sum_{k=0}^{n}\binom{n}{k}\left|k-\frac{n}{2}\right| = \begin{cases}\frac{n!!}{2(n-1)!!} & \text{, if } n \text{ odd}\\ \frac{(n-1)!!}{2(n-2)!!} & \text{, if } n \text{ even}\end{cases}$$

where $n!!$ is a double factorial, i.e.

$$n!! \equiv \begin{cases} n\cdot(n-2)\cdot\ldots\cdot 5\cdot 3\cdot 1 = \frac{(n+1)!}{2^{(n+1)/2}(\frac{n+1}{2})!} & \text{, if } n>0 \text{ is odd}\\ n\cdot(n-2)\cdot\ldots\cdot 6\cdot 4\cdot 2 = 2^{n/2}(\frac{n}{2})! & \text{, if } n>0 \text{ is even}\\ 1 & \text{, if } n=-1,0\end{cases}$$

According to Stirling's approximation[2] we have

$$\sqrt{2\pi} n^{n+1/2} e^{-n+1/(12n+1)} < n! < \sqrt{2\pi} n^{n+1/2} e^{-n+1/(12n)} \tag{3.3}$$

After some calculations, the following conjecture emerges, whose correctness can easily be verified:

$$\mathrm{MD} = 2^{-n} \sum_{k=0}^{n} \binom{n}{k} \left| k - \frac{n}{2} \right| \leq \sqrt{\frac{n}{\pi}} \ .$$

□

However, note that these results do not influence our asymptotic bounds. Hence, in the following, we will not consider random rounding.

3.1.3 Information Aggregation

Our second component is an information aggregation algorithm. Distributed aggregation is an interesting field of research, and there exist many papers on the subject [129, 136, 167, 186] (cf also the excellent introductory book by Peleg [187]). In the following, we concentrate on aggregation on hypercubes.

When the total number of peers in the d-dimensional hypercube system exceeds a certain threshold, all nodes $\beta_0 \dots \beta_{d-1}$ have to split into two new nodes $\beta_0 \dots \beta_{d-1}0$ and $\beta_0 \dots \beta_{d-1}1$, yielding a $(d+1)$-dimensional hypercube. Analogously, if the number of peers falls beyond a certain threshold, nodes $\beta_0 \dots \beta_{d-2}0$ and $\beta_0 \dots \beta_{d-2}1$ have to merge their peers into a single node $\beta_0 \dots \beta_{d-2}$, yielding a $(d-1)$-dimensional hypercube. Based on ideas also used in [13, 235, 236], we present an algorithm which provides the same estimated number of peers in the system to all nodes in every step allowing all nodes to split or merge synchronously, that is, in the same step. The description is again made in terms of *tokens* rather than peers.

Assume that in order to compute the total number of tokens in a d-dimensional hypercube, each node $v = \beta_0 ... \beta_{d-1}$ maintains an array $\Gamma_v[0...d]$, where $\Gamma_v[i]$ for $i \in [0, d]$ stores the estimated number of tokens in the sub-cube consisting of the nodes sharing v's prefix $\beta_0 ... \beta_{d-1-i}$. Further, assume that at the beginning of each step, an adversary inserts and removes an arbitrary number of tokens at arbitrary nodes. Each node $v = \beta_0 ... \beta_{d-1-i} ... \beta_{d-1}$ then calculates the new array $\Gamma'_v[0...d]$. For this, v sends $\Gamma_v[i]$ to its adjacent node $u = \beta_0 ... \overline{\beta_{d-1-i}} ... \beta_{d-1}$, for $i \in [0, d-1]$. Then, $\Gamma'_v[0]$ is set to the new number of tokens at v which is the only node with prefix $\beta_0 ... \beta_{d-1}$. For $i \in [1, d]$, the new estimated number of tokens in the prefix domain $\beta_0 ... \beta_{d-1-(i+1)}$ is given by the total number of tokens in the domain $\beta_0 ... \beta_{d-1-i}$ plus the total number of tokens in domain $\beta_0 ... \overline{\beta_{d-1-i}}$ provided by node u, that is, $\Gamma'_v[i+1] = \Gamma_v[i] + \Gamma_u[i]$.

[2]The double Inequality (3.3) is actually an extended version of Stirling's approximation.

Lemma 3.6. *Consider two arbitrary nodes v_1 and v_2 of the d-dimensional hypercube. The algorithm guarantees that $\Gamma_{v_1}[d] = \Gamma_{v_2}[d]$ at all times t. Moreover, it holds that this value is the correct total number of tokens in the system at time $t - d$.*

Proof. We prove by induction that at time $t + k$, all nodes v sharing the prefix $\beta_0 ... \beta_{d-1-k}$ for $k \in [0, d]$ store the same value $\Gamma_v[k]$ which represents the correct state of that sub-domain in step t.

$k = 0$: There is only one node having the prefix $\beta_0 ... \beta_{d-1}$, so the claim trivially holds.

$k \rightarrow k+1$: By the induction hypothesis, nodes v with prefix $\beta_0 ... \beta_{d-1-(k+1)}$ β_{d-1-k} share the same value $\Gamma_v[k]$ corresponding to the state of the system k steps earlier; the same holds for all nodes u with prefix $\beta_0 ... \beta_{d-1-(k+1)} \overline{\beta_{d-1-k}}$. In step $k+1$, all these nodes having the same prefix $\beta_0 ... \beta_{d-1-(k+1)}$ obviously store the value $\Gamma'_v[k+1] = \Gamma'_u[k+1] = \Gamma_v[k] + \Gamma_u[k]$. □

3.2 The Dynamic Hypercube System

Based on the components presented in the previous sections, both the topology and the maintenance algorithm are now described in detail. In particular, we show that, given an adversary $\mathcal{A}(d+1, d+1, 6)$ who inserts and removes at most $d + 1$ peers in any time interval of 6 rounds, 1) the out-degree of every peer is bounded by $\Theta(\log^2 n)$ where n is the total number of peers in the system, 2) the network diameter is bounded by $\Theta(\log n)$, and 3) every node of the simulated hypercube has always at least one peer which stores its data items, and hence no data item will ever be lost.

3.2.1 Topology

We start with a description of the overlay topology. As already mentioned, the peers are organized to simulate a d-dimensional hypercube, where the hypercube's nodes are represented by a group of peers. A data item with identifier ID is stored at the node whose identifier matches the first d bits of the hash-value of ID.

The peers of each node v are divided into a *core* $\mathcal{C}_v$ of at most $2d + 3$ peers and a *periphery* $\mathcal{P}_v$ consisting of the remaining peers; all peers within the same node are completely connected (*intra-connections*). Moreover, every peer is connected to all *core* peers of the neighboring nodes (*inter-connections*). Figure 3.1 shows an example for $d = 2$.

The data items belonging to node v are replicated on all core peers, while the peripheral peers are used for the balancing between the nodes according to the peer distribution algorithm and do not store any data items. The partition into core and periphery has the advantage that the peers which move between nodes do not have to replace the data of the old node by the data of the new node in most cases.

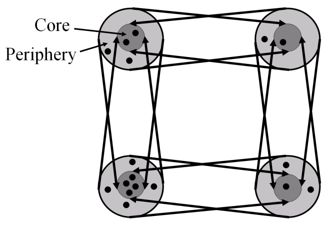


Figure 3.1: A simulated 2-dimensional hypercube with four nodes, each consisting of a core and a periphery. All peers within the same node are completely connected to each other, and additionally, all peers of a node are connected to all core peers of the neighboring nodes. Only the core peers store data items, while the peripheral peers move between the nodes to balance biased adversarial changes.

3.2.2 6-Round (Maintenance) Algorithm

The *6-round (maintenance) algorithm* maintains the simulated hypercube topology described in the previous section given an adversary $\mathcal{A}(d+1, d+1, 6)$. In particular, it ensures that 1) every node has at least one core peer all the time, and hence no data is lost; 2) each node has between $3d + 10$ and $45d + 86$ peers; 3) only peripheral peers are moved between nodes, thus the unnecessary copying of data is avoided.

In the following, we refer to a complete execution of the six rounds (Round 1 – Round 6) of the maintenance algorithm as a *phase*. Basically, the 6-round algorithm balances the peers across one dimension in every phase according to the token distribution algorithm as described in Chapter 3.1.1; additionally, the total number of peers in the system is computed with respect to an earlier state of the system by the information aggregation algorithm of Chapter 3.1.3 to expand or shrink the hypercube if the total number of peers exceeds or falls below a certain threshold. In our system, we use the lower threshold $LT = 8d + 16$ and the upper threshold $UT = 40d + 80$ for the total number of peers *per node on average.*[3]

While peers join and leave the system at arbitrary times, the 6-round algorithm considers the (accumulated) changes only once per phase. That is, a snapshot of the system is made in Round 1; Rounds 2 – 6 then ignore the changes that might have happened in the meantime and depend solely on the snapshot at the beginning of the phase.

[3]Note that since we consider the threshold *on average*, and since these values are provided with a delay of d phases in a d-dimensional hypercube (see Lemma 3.6), the number of peers at an individual node can lie outside the threshold interval.

Round 1

Outline: Each node v makes a snapshot of the currently active peers, denoted by the ID set $\mathcal{S}_v$. The later rounds will only be based on these sets.
Sent Messages: Each peer of a node v sends a packet with its own ID and the (potentially empty) ID set of its joiners to all adjacent peers *within* v.

Round 2

Outline: Based on the snapshot of Round 1, the core peers of a node v know the total number of peers in the node, $size(v) = |\mathcal{S}_v|$. This information is needed for the peer distribution algorithm and for the estimation of the total number of peers in the system.
Local Computations: The core peers compute $size(v) = |\mathcal{S}_v|$.
Sent Messages: Each peer informs its joiners about $\mathcal{S}_v$. The core peers $\mathcal{C}_v$ additionally send the number $size(v)$ to their neighboring core $\mathcal{C}_u$, where node u is v's neighbor in dimension i—the node with which v has to balance its peers in this phase. The core also exchanges the new estimated total number of peers in its domains with the corresponding adjacent cores.

Round 3

Outline: At the beginning of this round, every peer within a node v knows $\mathcal{S}_v$, and the transfer for the peer distribution algorithm can be prepared. Let v again be an arbitrary node and u its adjacent node in dimension i. We assume that $size(v) > size(u)$; the case where $size(v) \leq size(u)$ is analogous and not described further here. The ID set $\mathcal{T}$ of peers that have to move from node v to node u are the $(size(v) - size(u))/2$ (arbitrarily rounded) peers in the periphery $\mathcal{P}_v$ having the smallest identifiers.
Local Computations: The peers in each node v compute the new periphery $\mathcal{P}_v = \mathcal{S}_v \setminus \mathcal{C}_v$. The core remains the same.
Sent Messages: All cores forward the information about the new estimated total number of peers in the system to their peripheral peers. Moreover, the core of the larger node $\mathcal{C}_v$ sends the identifiers of the to be transferred peers $\mathcal{T}$ to $\mathcal{C}_u$, and the number $(size(v) - size(u))/2$ to the new periphery $\mathcal{P}_v$.

Round 4

Outline: The transfer for the peer distribution algorithm is continued. Moreover, this round prepares the dimension reduction if necessary.
Sent Messages: The core $\mathcal{C}_u$ informs the peers in $\mathcal{T}$ about all neighboring cores $\mathcal{C}_{u_j}$, where u_j is the neighbor of u in dimension j for $j \in [0, d-1]$, about $\mathcal{C}_u$ itself, about $\mathcal{S}_u$ and about its peripheral peers $\mathcal{P}_u$. Additionally, $\mathcal{C}_u$ informs its own periphery $\mathcal{P}_u$ about the newcomers $\mathcal{T}$.

If the estimated total number of peers in the system is beyond the threshold, the core peers of a node $\overline{v}$ which will be reduced send their data items plus the identifiers of all their peripheral peers (with respect to the situation *after* the transfer) to the core of their adjacent node $\underline{v}$.

Round 5

Outline: This round finishes the peer distribution, establishes the new peripheries, and prepares the building of a new core. If the hypercube has to grow in this phase, the nodes start to split, and vice versa if the hypercube is going to shrink.
Local Computations: Given the number $(size(v) - size(u))/2$, the peripheral peers $\mathcal{P}_v$ can compute the set $\mathcal{T}$ selecting the $(size(v) - size(u))/2$ smallest elements in $\mathcal{P}_v$. From this, the new periphery $\mathcal{P}_v = \mathcal{P}_v \setminus \mathcal{T}$ is computed. Analogously, the peers in node u (including $\mathcal{T}$) can compute the new periphery $\mathcal{P}_u = \mathcal{P}_u \cup \mathcal{T}$.

Then, all peers of each node v calculate the new core $\mathcal{C}^{new}_v$: it consists of the peers of the old core which have still been alive in Round 1, i.e., $\mathcal{C}^{old}_v = \mathcal{C}_v \cap \mathcal{S}_v$, plus the $2d+3-|\mathcal{C}_v \cap \mathcal{S}_v|$ smallest IDs in the new periphery $\mathcal{P}_v$, denoted by $\mathcal{C}^{\triangle}_v$. Hence, the new core is given by $\mathcal{C}^{new}_v = \mathcal{C}^{old}_v \cup \mathcal{C}^{\triangle}_v$, and the new periphery by $\mathcal{P}^{new}_v = \mathcal{P}_v \setminus \mathcal{C}^{\triangle}_v$.

If the hypercube has to grow in this phase, the smallest $2d+3$ peers in the new periphery $\mathcal{P}^{new}_{\underline{v}}$ become the new core of the expanded node, $\mathcal{C}_{\overline{v}}$. Half of the remaining peripheral peers, the ones with the smaller identifiers, build the new periphery $\mathcal{P}_{\overline{v}}$, and the other half becomes $\mathcal{P}_{\underline{v}}$. All these operations can be computed locally by every peer.
Sent Messages: The old core $\mathcal{C}^{old}_v$ informs all its neighboring nodes (i.e., their old cores) about the new core $\mathcal{C}^{new}_v$. Moreover, $\mathcal{C}^{old}_v$ sends its data items to the peers in $\mathcal{C}^{\triangle}_v$.

If the hypercube is about to grow, $\mathcal{C}^{old}_{\underline{v}}$ sends the necessary data items to the core peers of the new node, $\mathcal{C}_{\overline{v}}$. Moreover, $\mathcal{C}^{old}_{\underline{v}}$ informs its neighboring (old) cores about the IDs of its expanded core $\mathcal{C}_{\overline{v}}$.

If the hypercube is about to shrink, all cores $\mathcal{C}^{old}_{\underline{v}}$ inform their periphery about the peers arriving from the expanded node and the peers in the expanded node about the new core $\mathcal{C}^{new}_{\underline{v}}$ and its periphery. $\mathcal{C}^{old}_{\underline{v}}$ also copies the data items of $\mathcal{C}^{old}_{\overline{v}}$ to the peers $\mathcal{C}^{\triangle}_{\underline{v}}$.

Round 6

Outline: The new cores are built and the dimension change is accomplished if necessary.
Local Computations: If the hypercube has been reduced, every peer can now compute the new periphery $\mathcal{P}_{\underline{v}}$.
Sent Messages: The old core $\mathcal{C}^{old}_v$ forwards the information about the new neighboring cores to the peers $\mathcal{C}^{\triangle}_v \cup \mathcal{P}_v$.

If the hypercube has grown, $\mathcal{C}^{old}_{\underline{v}}$ forwards the expanded cores of its neighboring nodes to *all* peers in its expanded node $\overline{v}$. Note that this requires that $\mathcal{C}^{old}_{\underline{v}}$ remembers the peripheral peers that have been transferred to $\overline{v}$ in Round 5.

Theorem 3.7. *Given an adversary $\mathcal{A}(d+1, d+1, 6)$ who inserts and removes at most $d+1$ peers per phase, the described 6-round algorithm ensures that 1) every node always has at least one core peer and hence no data is lost;*

2) each node has between $3d + 10$ and $45d + 86$ peers, yielding a logarithmic network diameter; 3) only peripheral peers are moved between nodes, thus the unnecessary copying of data is avoided.

Proof. We first consider a simpler system without the separation into core and periphery, where the maintenance algorithm simply runs the peer distribution algorithm and the information aggregation algorithm to count the total number of peers in the system, and expands or reduces the hypercube with respect to the thresholds $LT = 8d + 16$ and $UT = 40d + 80$. Moreover, we assume that these operations are performed in quiet phases, where the adversary removes at most $d+1$ and adds at most $d+1$ peers only in-between.

For this simpler system, it holds that every node in the simulated d-dimensional hypercube has at least $3d + 10$ and at most $45d + 86$ peers at every moment of time. Moreover, after the hypercube has changed its dimension from d_{old} to d_{new}, the dimension will remain stable for at least $2d_{new} + 1$ phases.

The cases where the average number of peers per node μ falls beyond the lower threshold $8d_{old} + 16$ or exceeds the upper threshold $40d_{old} + 80$ are studied in turn. According to Lemma 3.6, such an event will lead to a dimension change with a delay of d_{old} phases only. We prove that after the change, $\mu \in [8d_{new} + 16, 40d_{new} + 80]$ for at least $d_{new} + 1$ phases. The dimension remains stable for at least $2d_{new} + 1$ phases which implies—together with Lemma 3.3—that the discrepancy before the next change is limited by $2(d_{new} + 1) + 2(d_{new} + 1) + d_{new} = 5d_{new} + 4$.

Case $\mu < 8d + 16$: At time $t - d_{old}$, it held that $\mu < 8d_{old} + 16$ while at time $t - d_{old} - 1$ we had $\mu \geq 8d_{old} + 16$. In $d_{old} + 1$ phases, there are at most $(d_{old}+1)(d_{old}+1) = d_{old}^2 + 2d_{old} + 1$ leaves, so $\mu \geq 8d_{old} + 16 - \frac{d_{old}^2 + 2d_{old} + 1}{2^{d_{old}}} > 8d_{old} + 14$ before merging. Clearly, there must be a node with more than $8d_{old} + 14$ peers, hence, given the discrepancy of $5d_{old} + 4$ (cf Lemma 3.3), every node has more than $3d_{old} + 10$ peers before merging.

What about the maximum? At time $t - d_{old}$, $\mu < 8d_{old} + 16$, and there have been at most $d_{old}(d_{old}+1)$ joins in d_{old} steps, so $\mu < 8d_{old} + 16 + \frac{d_{old}(d_{old}+1)}{2^{d_{old}}} < 8d_{old} + 18$ before merging, and $\mu < 16d_{old} + 36$ afterwards. The maximum node has less than $21d_{new} + 61$ peers.

We now show that $\mu \geq 8d_{new} + 16$ for the next $d_{new} + 1$ phases after a reduction. At time $t - d_{old} - 1$, $\mu \geq 8d_{old} + 16 = 8d_{new} + 24$. The reduction doubles the average number of peers per node, hence $\mu \geq 16d_{new} + 48$. Further, there are at most $(d_{old}+1)(d_{old}+1) + (d_{new}+1)(d_{new}+1) = 2d_{new}^2 + 6d_{new} + 5$ leaves in the meantime, hence $\mu \geq 16d_{new} + 48 - \frac{2d_{new}^2 + 6d_{new} + 5}{2^{d_{new}}} > 16d_{new} + 41 > 8d_{new} + 16$.

Finally, $\mu \leq 40d_{new} + 80$ for $d_{new} + 1$ phases. At time $t - d_{old}$, $\mu < 8d_{new} + 24$, so $\mu < 16d_{new} + 48$ after the reduction. There are at most $d_{old}(d_{old} + 1) + (d_{new} + 1)(d_{new} + 1) = 2d_{new}^2 + 5d_{new} + 3$ joins, therefore $\mu < 16d_{new} + 48 + \frac{2d_{new}^2 + 5d_{new} + 3}{2^{d_{new}}} < 16d_{new} + 54 < 40d_{new} + 80$.

Case $\mu > 40d + 80$: At time $t - d_{old}$, $\mu > 40d_{old} + 80 = 40d_{new} + 40$, so $\mu > 20d_{new} + 20$ after splitting; there are at most $d_{old}(d_{old}+1) = d_{new}^2 - d_{new}$

leaves in d_{old} steps, so $\mu > 20d_{new}+20-\frac{d_{new}^2-d_{new}}{2d_{new}} > 20d_{new}+19$. According to Lemma 3.3, the minimum node has more than $15d_{new}+15$ peers after splitting. At time $t-d_{old}-1$, $\mu \leq 40d_{old}+80$, and there are at most $(d_{old}+1)(d_{old}+1) = d_{old}^2+2d_{old}+1$ joins. Hence, before splitting, $\mu \leq 40d_{old}+80+\frac{d_{old}^2+2d_{old}+1}{2d_{old}} < 40d_{old}+82$, and the maximum node has at most $45d_{old}+86$ peers.

Next, we show that $\mu \geq 8d_{new}+16$ for the $d_{new}+1$ phases after the expansion. At time $t-d_{old}$, $\mu > 40d_{old}+80 = 40d_{new}+40$, hence $\mu > 20d_{new}+20$ after the expansion. Moreover, there are at most $d_{old}(d_{old}+1)+(d_{new}+1)(d_{new}+1) = 2d_{new}^2+d_{new}+1$ leaves, and $\mu > 20d_{new}+20-\frac{2d_{new}^2+d_{new}+1}{2d_{new}} > 20d_{new}+17 \geq 8d_{new}+16$. Finally, $\mu \leq 40d_{new}+80$ for the next $d_{new}+1$ steps: at time $t-d_{old}-1$, $\mu \leq 40d_{old}+80 = 40d_{new}+40$, so $\mu \leq 20d_{new}+20$ after the expansion; moreover, there are at most $(d_{old}+1)(d_{old}+1)+(d_{new}+1)(d_{new}+1) = 2d_{new}^2+2d_{new}+1$ joins, thus $\mu \leq 20d_{new}+20+\frac{2d_{new}^2+2d_{new}+1}{2d_{new}} < 20d_{new}+24 < 40d_{new}+80$.

In reality, the repairing operations will run *concurrently* to the adversary. However, as all operations are based on the state of Round 1, a phase can be considered as running uninterruptedly, that is, as if the adversary inserted $d+1$ and removed $d+1$ peers only *between* the phases. Thus, the properties shown above remain valid. However, we additionally have to postulate that there is always at least one *core peer*. We know that it is always possible to select $2d+3$ core peers in Round 5 with respect to the state of Round 1. These peers have to survive until Round 6 of the next phase, so for twelve normal rounds in total; however, as the adversary $\mathcal{A}_{adv}(d+1, d+1, 6)$ removes at most $2d+2$ peers in twelve rounds, this clearly holds.

Finally, we show that there are indeed enough peripheral peers in Round 3 such that core peers do not have to change the node for the peer distribution, that is: in Round 3, it holds that $|\mathcal{P}_v| > \frac{size(v)-size(u)}{2}$. We know that $size(v) \geq 3d+10$ and $size(u) \geq 3d+10$. As v has at most $2d+3$ core peers, we have $|\mathcal{P}_v| \geq size(v)-(2d+3) \geq size(v)-size(u) > \frac{size(v)-size(u)}{2}$. □

Finally, observe that we can replace the complete bipartite graphs between adjacent hypercube nodes by bipartite matchings, reducing the peer degree from $O(\log^2 n)$ to $O(\log n)$. Apart from the lower degrees, all our results still hold up to constant factors.

3.3 The Dynamic Pancake System

The previous section has presented algorithms to maintain a hypercube topology with diameter $O(\log n)$ and where each peer has at most $O(\log n)$ neighbors. The topology is resilient to $O(\log n)$ worst-case changes per time unit which is asymptotically optimal. In this section we sketch how our construction can be adapted for another interesting network topology, namely for the *pancake graph* of order d defined in Definition 3.1. The pancake graph has

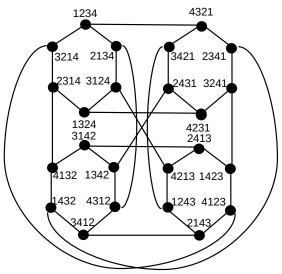


Figure 3.2: A pancake graph of order 4 (P_4).

the interesting property that it is a graph with minimal maximum of peer degree and network diameter.

Definition 3.1. *A* pancake graph *of order d is a graph $P_d = (V, E)$, with $V(P_d) = \{l_1 l_2 ... l_d \mid l_i \in \{1, ..., d\}, \forall i \neq j : l_i \neq l_j\}$, i.e., $V(P_d)$ is the set of all permutations on the set $[1, d]$. Let ρ_i denote a prefix-inversion of length i: $\rho_i(l_1 ... l_i ... l_d) = l_i l_{i-1} ... l_1 l_{i+1} ... l_d$. For $u, v \in V(P_d)$, it holds that $\{u, v\} \in E(P_d) \Leftrightarrow v = \rho_i(u)$ for $i \in \{2, ..., d\}$. P_d is a $(d-1)$-regular graph of diameter smaller than $2d$.*

Henceforth, we will refer to the set $\{a, a+1, ..., b-1, b\}$ as $[a, b]$. Moreover, the number l_i at the i^{th} position of a node with label $v = l_1 ... l_d$ will be called the i^{th} *entry.*

Again, we *simulate* the pancake graph in our p2p system: each peer is part of a distinct pancake node, and each pancake node consists of $O(d^2)$ peers. A data item is redundantly stored by the peers of the node to which its identifier hashes. Peers have connections to other peers of their pancake node; additionally, some peers of neighboring pancake nodes are connected to each other. In case of joins or leaves, some of the peers have to change to another pancake node such that up to constant factors, all pancake nodes own the same number of peers at all times. If the total number of peers grows or shrinks above or below a certain threshold, the order of the pancake is increased or decreased by one, respectively.

The balancing of peers among the pancake nodes is again done by a dynamic token distribution algorithm. Moreover, we have a distributed information aggregation algorithm which estimates the number of peers in the system and adapts the order accordingly. Based on the described structure, we get a p2p system with peer degree and network diameter $O(\log n / \log \log n)$, implying time complexity $O(\log n / \log \log n)$ for the usual operations such as search. At the same time, our system tolerates $\Theta(\log n / \log \log n)$ worst-case joins and/or crashes per constant time interval.

3.3.1 Dimension Change

The order of the pancake graph is changed according to the total number of peers in the system. For the expansion, node $l_1...l_d \in V(P_d)$ splits into $d+1$ new nodes $\{(d+1)l_1l_2...l_d, l_1(d+1)l_2...l_d, ..., l_1l_2...l_d(d+1)\}$ of P_{d+1}, and vice versa for the reduction.

To be useful for our application, the pancake's order change from d_{old} to d_{new} has to fulfill a requirement: a node in $P_{d_{new}}$ must be able to compute its new neighbors *locally*, i.e., based on the information about the neighbors in $P_{d_{old}}$. We will now describe the expansion and the reduction of the order in turn and show that this criterion is indeed met in both cases.

Expansion

If the total number of peers in the system exceeds a certain threshold, each node $v = l_1...l_d \in V(P_d)$ splits into $d+1$ new nodes $\{v^{exp}_{(1)} = (d+1)l_1l_2...l_d, v^{exp}_{(2)} = l_1(d+1)l_2...l_d, ..., v^{exp}_{(d+1)} = l_1l_2...l_d(d+1)\}$ of P_{d+1}. The following lemma states that the new neighbors of a node $v^{exp}_{(i)} \in V(P_{d+1})$ can easily be computed given the knowledge about the neighbors of the original node $v \in V(P_d)$.

Lemma 3.8. *Consider two arbitrary nodes u and v of P_d. It holds that if $\{u^{exp}_{(i)}, v^{exp}_{(j)}\} \in E(P_{d+1})$ for some $i, j \in \{1, ..., d+1\}$, then $\{u, v\} \in E(P_d)$ or $u = v$.*

Proof. If $\{u^{exp}_{(i)}, v^{exp}_{(j)}\} \in E(P_{d+1})$ there is a $k \in \{2, ..., d+1\}$ such that $u^{exp}_{(i)} = \rho_k(v^{exp}_{(j)})$. If $d+1$ appears among the first k entries of $u^{exp}_{(i)}$ (and thus also of $v^{exp}_{(i)}$), the original nodes—having no entry $(d+1)$—are related by a prefix-inversion of length $k-1$: $u = \rho_{k-1}(v)$. If on the other hand the entry $(d+1)$ appears among the remaining entries, u and v are related by the same prefix-inversion: $u = \rho_k(v)$. □

Reduction

If the total number of peers in the system falls below a certain threshold, all nodes $l_1...l_i(d+1)l_{i+1}...l_d \in V(P_{d+1})$ for $i \in [0, d]$ merge into a single node $l_1...l_d \in V(P_d)$. Unfortunately, we cannot reverse the expansion directly. Instead, the reduction works as follows. First, the following Dominating Set [101] on P_{d+1} is computed: every node $v = l_1...l_{d+1}$ having $l_1 = d+1$ becomes a dominator. We will call a dominator plus its adjacent (dominated) nodes a *cluster*. In the following, let $v^{dom}_{(1)} = (d+1)l_1...l_d$ be a dominator and $v^{dom}_{(i)} = \rho_i(v^{dom}_{(1)}) = l_{i-1}l_{i-2}...(d+1)l_i...l_d$ its neighbor with prefix-inversion of length i, for $i \in [1, d+1]$. The idea is to contract each cluster with dominator $v^{dom}_{(1)} = (d+1)l_1...l_d$ to a single node $v = l_1...l_d \in V(P_d)$. Mind, however, that our clusters do not yield the desired reduction yet: in order to get the inverse operation of the expansion, each cluster has to exchange one dominated node with each of its adjacent clusters.

Before we explain the exchange of the dominated nodes in detail, we first prove that the set of nodes having $l_1 = d+1$ indeed forms a dominating set, that every dominated node is adjacent to exactly one dominator, and that dominators are independent.

Lemma 3.9. *Consider the graph P_{d+1}. The $d!$ nodes of P_{d+1} with first entry $l_1 = d+1$ build a dominating set, i.e., each node is either a dominator itself or adjacent to a dominator. Moreover, clusters are disjoint.*

Proof. Consider an arbitrary node $v = l_1 l_2 ... l_{d+1}$. Assume that $l_i = d+1$ for some $i \in \{1, ..., d+1\}$. If $i = 1$, v is a dominator itself. Two nodes having $l_1 = d+1$ cannot be adjacent because of the prefix-inversion changes the first entry. If $i \neq 1$, there is exactly one neighbor of v which is a dominator, namely node $u = \rho_i(v)$. □

According to Lemma 3.9, each node belongs to exactly one cluster, hence the contraction operation is well-defined. However, as already mentioned, we additionally need to exchange dominated nodes between adjacent clusters. This is done as follows: the cluster with dominator $v^{dom}_{(1)} = (d+1)l_1...l_d$ sends its dominated node $v^{dom}_{(i+1)}$ to the cluster with dominator $(d+1)\rho_i(l_1...l_d)$, for $i \in [2, d]$.

It holds that after the exchange of the dominated nodes, (i) each cluster with dominator $v^{dom}_{(1)} = v^{exp}_{(1)} = (d+1)l_1...l_d$ which will contract to node $v = l_1...l_d$ consists of the nodes $v^{exp}_{(1)} = (d+1)l_1...l_d, v^{exp}_{(2)} = l_1(d+1)...l_d, ..., v^{exp}_{(d+1)} = l_1...l_d(d+1)$, and (ii) the dominated node $v^{exp}_{(i)}$ for $i \in [3, d+1]$—before being transferred to the cluster dominated by $v^{dom}_{(1)}$—belonged to the cluster that will form the new node $\rho_i(v)$. To see this, note that node $v^{dom}_{(i)}$ is replaced by $\rho_{i-1}(v^{dom}_{(i)}) = \rho_{i-1}(l_{i-1}...l_1(d+1)l_i...l_d) = v^{exp}_{(i)}$, and that before the transfer, $v^{exp}_{(i)}$ belonged to the cluster dominated by $\rho_i(v^{exp}_{(i)}) = (d+1)l_{i-1}...l_1 l_i...l_d$ which will reduce to node $\rho_{i-1}(v)$. Thus, after the exchange, the following lemma holds.

Lemma 3.10. *The cluster contracting to node v consists of those nodes which v would also expand to, and the cluster has information about each of v's neighbors.*

3.3.2 Information Aggregation

Our algorithm $\mathcal{A}_{IA}$ allows to count the total number of peers in the pancake's nodes. Let $P_i(v)$ denote the sub-graph of the pancake graph P_d consisting of those nodes which share a postfix of length $d-i$ with a given node v. (Note that the graph induced by $P_i(v)$ is a pancake graph of order i.) The algorithm runs in $d-1$ phases and accumulates the total number of tokens in sub-graphs of increasing size.

Each phase consists of two rounds. In the first round of phase i, a node v sends the total number of tokens in its sub-graph $P_i(v)$—which is known by induction—to its neighbor $\rho_{i+1}(v)$. Thus, since prefix-inversion is a symmetric operation, v receives the total number of tokens in the sub-graph $P_i(\rho_{i+1}(v))$ from node $\rho_{i+1}(v)$. In the second round, node v sends this information to all neighbors $\rho_j(v)$ for $j < i+1$. Given the information about all $P_i(\rho_{i+1}(\rho_j(v)))$ (for $j < i+1$), the total number of tokens in the sub-graph $P_{i+1}(v)$ can be computed: $\tau(P_{i+1}(v)) = \tau(P_i(v)) + \sum_{j=1}^{i} \tau(P_i(\rho_{i+1}(\rho_j(v))))$, where $\tau(\cdot)$ denotes the number of tokens in the corresponding sub-graph. Hence, by induction, after $d-1$ phases, every node can compute the total number of tokens in the system.

Theorem 3.11. *$\mathcal{A}_{IA}$ provides all nodes with the correct total number of tokens in the system after $d-1$ phases.*

Proof. By induction over the phases we show that after phase i, it holds that each node v knows the total number of tokens in $P_{i+1}(v)$.

$i = 0$: Before the first phase, a node v only knows its own tokens, and as there is only one node in $P_1(v)$, the claim holds trivially.

$i \to i+1$: By the induction hypothesis, after phase i, each node $v = l_1...l_d$ knows the total number of tokens in the sub-graph $P_{i+1}(v)$. In phase $i+1$, node v learns the total number of tokens in the sub-graphs $P_{i+1}(\rho_{i+2}(\rho_j(v)))$ for $j < i+2$. This facilitates the computation of the total number of tokens in $P_{i+2}(v)$.

Note that the nodes $\rho_j(v)$ for $j < i+2$ all have a different first entry and share the postfix $l_{i+2}l_{i+3}...l_d$ with v. Performing a ρ_{i+2} prefix-inversion yields a member for each sub-graph with postfix $l_{i+3}l_{i+4}...l_d$ of length $d-(i+2)$. Therefore, combining the information of the sub-graphs gives the total number of tokens in $P_{i+2}(v)$. □

$\mathcal{A}_{IA}$ is executed all the time and in a pipelined fashion, i.e., all phases run concurrently. This way, all nodes always get a consistent result even if the adversary concurrently adds and removes tokens (peers). Moreover, the result always corresponds to the exact state of the system $d-1$ phases ago.

3.3.3 Token Distribution

Our goal is again to minimize the maximum difference of the number of tokens of any two pancake nodes, denoted by the *discrepancy* ϕ. Analogously to the information aggregation algorithm, our token distribution algorithm $\mathcal{A}_{TD}$ exploits the recursive structure of the pancake graph. In a first step, all pancakes of order 2 balance their tokens. Then, the pancakes of order $3, 4, \ldots$ exchange tokens. Pancakes of order i can thereby build on the fact that all pancakes of order $i-1$ have balanced the token levels of their nodes. A detailed description of $\mathcal{A}_{TD}$ is given in Algorithm 3.1. We assume that we have a dominating set for each pancake $P_i(v)$. For example, the dominators could again be all nodes of $P_i(v)$ having the largest of the first i entries at the first position. Note that, by definition, entries $i+1$ to d are fixed for all nodes of $P_i(v)$.

```
1: for $i := 2$ to $d$ do
2:     send all tokens to $\rho_i(v)$;
3:     send all tokens to dominator in $P_i(v)$;
4:     dominators send tokens to nodes of their clusters;
5: end for
```

Algorithm 3.1: Token Distribution $\mathcal{A}_{TD}$ (node v)

Let $P_i(v)$ be the pancake of order i. After the i^{th} iteration of $\mathcal{A}_{TD}$, for all v, all nodes of $P_i(v)$ have the same number of tokens. Hence, at the end ($i = d$) all nodes of the pancake have the same number of tokens. In Line 4 of $\mathcal{A}_{TD}$, it is not specified how many tokens to send to which nodes if the number of tokens at a node is not divisible by i. There is also no explicit notion of tokens which are added or removed by an adversary during the algorithm. In the following, we will prove that the algorithm perfectly distributes tokens if tokens are fractional, that is, if they can be divided arbitrarily and if no tokens are added or removed during the algorithm (static token distribution). We will then analyze the effects of adversarial insertions and deletions and of integer tokens.

Lemma 3.12. *$\mathcal{A}_{TD}$ perfectly solves the static fractional token distribution problem on a pancake of order d.*

Proof. As outlined above, we prove the lemma by induction over i. Since $P_1(v)$ is a single node, clearly at the beginning all nodes of $P_1(v)$ have the same number of tokens. Let us therefore assume that for all nodes u, each node of $P_{i-1}(u)$ has the same number of tokens $\tau_{i-1}(u)$. The pancakes $P_{i-1}(u)$ of order $i-1$ belonging to $P_i(v)$ can be characterized by their i^{th} entry. Let l_i be the i^{th} entry of the nodes of $P_{i-1}(u)$. In Line 2 of $\mathcal{A}_{TD}$, a node u of $P_{i-1}(u)$ moves all tokens to $\rho_i(u)$, that is, all tokens are moved to a node with l_i as its first entry. Hence, after Line 2, all nodes of $P_i(u)$ with first entry l_i have $\tau_{i-1}(u)$ tokens.

In Lines 3 and 4, each cluster (dominator plus neighbors) distributes all its tokens equally among the members of the cluster. It therefore remains to show that each cluster of $P_i(u)$ has the same number of tokens. However, since in each cluster, every possible first entry occurs exactly once, this is clear from the discussion of the first step of the algorithm (Line 2). □

We will now show how dynamic insertions and deletions of tokens affect the fractional token distribution of $\mathcal{A}_{TD}$. For the dynamic token distribution algorithm, we assume that the $d-1$ iterations of the algorithm are repeated, that is, after $i = d$, we start again at $i = 2$.

Lemma 3.13. *If in every iteration of $\mathcal{A}_{TD}$ at most J tokens are added and at most L tokens are removed, the algorithm guarantees that at all times $t \geq d-1$, the maximal difference between the numbers of fractional tokens between any two nodes is $3(J + L)$.*

Proof. To start, we only consider insertions and neglect deletions. Because all operations of the algorithm are linear, we can look at each token independently. By Lemma 3.12, each token which is added before the first iteration of the algorithm is distributed equally among $i! \geq 2^i$ nodes after iteration i. A token which is added after iteration j is distributed among $i!/j! \geq 2^{i-j}$ nodes after iteration i. All tokens which were inserted before the last complete execution of $\mathcal{A}_{TD}$ are equally distributed among all nodes of the pancake. We therefore only have to look at the last complete execution and at the current execution of the algorithm. All tokens which are inserted in the current execution of $\mathcal{A}_{TD}$ are distributed among at least 2^t nodes, t iterations after the insertion. Therefore, by a geometric series argument, there are at most $2J$ tokens per node which were inserted in the current iteration. All tokens which were inserted before the end of the last complete execution of the algorithm, were distributed among at least d nodes after the last complete execution. Since in iteration i, each node distributes its tokens among i different nodes and each node receives tokens from i different nodes, all the tokens from the last complete execution of the algorithm remain distributed among at least d nodes. Because there are at most $(d-1)J$ such tokens, each node has less than one of them. Together, the difference between the number of tokens at the heaviest and the lightest node becomes $3J$. For deleted tokens the same argumentation as for inserted tokens holds. □

We have analyzed the token distribution algorithm for the idealized case where tokens can be divided arbitrarily. In our application, tokens correspond to peers, and we again have to extend the analysis to integer tokens. We assume that in Line 4, tokens are distributed as equally as possible. That is, if there are k tokens in a cluster, some of the nodes receive $\lfloor k/i \rfloor$ tokens and some nodes receive $\lceil k/i \rceil$ tokens.

Lemma 3.14. *The (absolute) difference between the number of integer tokens and the number of fractional tokens at any node is always upper bounded by $2d$.*

Proof. We start the proof by looking at iteration i of $\mathcal{A}_{TD}$. Assume that before iteration i, the difference between the number of integer tokens and the number of fractional tokens is at most ξ at each node. If there are token insertions or deletions at a node, this difference does not change because insertions and deletions affect the numbers of fractional and integer tokens in the same way. In Line 2, all tokens are moved and therefore ξ remains unchanged. In Lines 3 and 4, tokens are distributed equally among i nodes of a cluster. If there are k tokens in such a cluster, each node gets between $\lfloor k/i \rfloor$ and $\lceil k/i \rceil$ tokens. If every node got exactly k/i tokens, the difference between fractional and integer would remain at most ξ. Due to the rounding, the difference can therefore grow to at most $\xi + 1$ after iteration i. Hence, after t iterations, the absolute difference between the numbers of fractional and integer tokens is at most t.

To prove that at each node, the number of integer tokens cannot deviate from the number of fractional tokens by more than $2d$, we need the following observation. By Lemma 3.12, fractional tokens are distributed equally among all nodes after their first complete execution of $\mathcal{A}_{TD}$, that is, after less than $2d$ iterations. Therefore, the number of fractional tokens at each node only depends on the insertions and deletions of the last $2d$ iterations and on the total number of tokens in the system. Therefore, the distribution of fractional tokens is the same if we assume that before the last $2d$ iterations, the number of fractional tokens at each node was equal to the number of integer tokens. By the above argumentation, the difference between the numbers of integer and fractional tokens at a node can have grown to at most $2d$ in those $2d$ iterations. □

By combining Lemmas 3.12, 3.13, and 3.14, we obtain the following theorem about the dynamic integer token distribution algorithm.

Theorem 3.15. *The discrepancy ϕ of the dynamic integer token distribution algorithm is at most $\phi \leq 4d + 3(J + L)$.*

The algorithm $\mathcal{A}_{TD}$ is formulated in the form which makes the proofs of this section as simple as possible. It is of course not desirable that all nodes first have to move all tokens to dominator nodes which then redistribute the tokens. Especially in the case where no insertions or deletions occur, we would like the system to stabilize to a point where no tokens have to be moved around. It is not difficult to implement $\mathcal{A}_{TD}$ in a way which has this property. In Line 2, two nodes u and $\rho_i(u)$ exchange all their tokens. They can of course obtain the same effect by computing the difference between the number of tokens and by only moving this number of tokens in the appropriate direction. A similar trick can be applied for Lines 3 and 4. The dominator nodes can collect all the necessary information and decide about the necessary movements of tokens.

3.3.4 Node Representation

The algorithms so far have all been described on the level of pancake graphs. In this section, we take a more detailed look at the internals of the system. We first present the representation of the pancake's nodes and edges and then give an algorithm which allows to maintain these structures against a concurrent adversary. We omit the peer-level description of some components, for example the token distribution or the information aggregation algorithm. These operations are straight-forward and can be done with similar techniques.

The Grid

The peers of a node $v \in V(P_d)$ are arranged to form a 2-dimensional grid G_v consisting of exactly $d + 1$ columns, while the number of rows R may vary depending on the total number of peers in v.

Let $\tau(v)$ be the total number of peers in node v and let $R = \lfloor \tau(v)/(d + 1) \rfloor$. The first $R \cdot (d + 1)$ peers are arranged in a 2-dimensional grid with $d + 1$ columns and R complete rows, such that every peer occupies exactly one

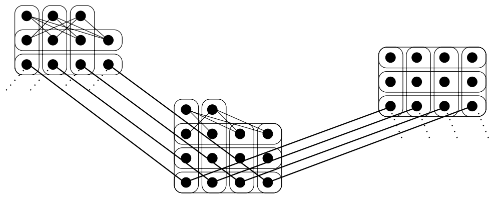


Figure 3.3: The peers of a pancake node are arranged as a grid with $d+1$ columns. A peer has connections to all peers in its row plus to all peers in its column. The pancake's edges are represented by a matching between the peers of the bottom row.

position $G_v[x,y]$ for $x \in [0,d]$ and $y \in [0,R-1]$. The remaining $\tau(v)$ *mod* $(d+1)$ peers—from now on called *extra peers*—are located in an incomplete additional row $G_v[i,R]$ for $i \in [0,\tau(v)\ mod\ (d+1)]$. Inside a row or column, the peers are completely connected ("intra-connections"): a peer at $G_v[x,y]$ is connected to the peers $G_v[x,i]$ for $i \in [0,R]$ and $G_v[i,y]$ for $i \in [0,d]$. As the extra peers do not form a complete row, they are more vulnerable; thus, they additionally participate in row $R-1$, i.e., we also have connections between $G_v[i,R]$ for $i \in [0,d+1]$ and *all* peers $G_v[j,R-1]$ for $j \in [0,\tau(v)$ *mod* $(d+1)]$.

Additionally, we need to specify the representation of the pancake's edges ("inter-connections"). The idea is as follows: if two nodes u and v are connected in the pancake graph P_d, i.e., $\{u,v\} \in E(P_d)$, then each peer $G_u[i,0]$ is connected to the peer occupying $G_v[i,0]$, for $i \in [0,d]$. In the following, we will call the peers in the lowest row (row 0) the *core* of the corresponding node. Thus, two nodes are connected by a *matching* between their cores. The representation of the pancake's nodes is depicted in Figure 3.3.

Grid Maintenance

Algorithm $\mathcal{A}_{GRID}$ needs several rounds. The main theme is similar to the one proposed for the hypercube graph: at the beginning, a snapshot of the state (living peers, etc.) of the system is made. The following rounds are then solely based on this information—ignoring the fact that some peers have been crashed by the concurrent adversary in the meantime. That is, by using sufficient redundancy, we do not have to take the crashed and newly joined peers into consideration until the maintenance algorithm restarts with the first round.

$\mathcal{A}_{GRID}$ consists of two phases. In the first phase, the following information is broadcast throughout the grid: (1) the positions where peers have left, (2) the IP addresses of the peers that have joined, (3) the IP addresses of the extra peers, and (4) the IP addresses of the peers in row $R-1$. The second phase is based on this information and works as follows: every surviving peer can locally compute which peers will take the positions of the peers that left (gaps in the grid). Thereby, newly joined peers are taken into account first, and if this is not enough, the extra peers are used. If there are still gaps in the grid, the peers of the top row are used, and if necessary, the number of rows is decremented ($R := R-1$). If on the other hand there are still joining peers left after all gaps have been filled, these peers are added to the top row, creating a new top row if necessary ($R := R + 1$). After this local computation, the peers that have to fill the gaps are provided with the information about their new neighbors. We can guarantee that no row is removed completely and that there is always a complete column in the presence of a concurrent adversary $\mathcal{A}_{ADV}(d/2, d/2, 5)$ who adds and removes at most $d/2$ peers in any time period of 5 rounds. Moreover, also the pancake's edges are repaired in constant time since we ensure that two adjacent pancake nodes always have at least two living adjacent core peers which can reestablish the matching between the cores.

We now give the detailed description of $\mathcal{A}_{GRID}$. We write $G_v[\cdot, y]$ and $G_v[x, \cdot]$ to denote all (surviving) peers in the y^{th} row and in the x^{th} column respectively. In the following, we assume the extra peers to participate in both rows R *and* $R-1$, i.e., they send and receive messages for both rows.
Round 1: The snapshot is made: a surviving peer at position $G_v[x, y]$ sends its IP address and the IP addresses of its joiners to all peers in $G_v[\cdot, y]$.
Round 2: Each peer at position $G_v[x, y]$ sends the addresses of its joiners plus the information in which column of its row peers have left to $G_v[x, \cdot]$.
Round 3: Each peer at position $G_v[x, y]$ forwards the information received in Round 2 to the peers $G_v[\cdot, y]$.
Round 4: Now the new form of G_v is computed locally: if a peer at $G_v[x, y]$ has missing neighbors on its row or column, it computes which joiner or—if necessary—which extra peer or which peer in the top row has to replace it. If there are enough new peers, the number of rows is incremented, and vice versa if more than all extra peers are used for repairing. Each peer having a missing neighbor on its row sends the information about all neighbors of this row or column directly to the peer which will replace it. Additionally, the information required to establish the top rows is provided to the responsible peers. Finally, in order to repair the matching between adjacent nodes, the peers of the old core which are still alive send the addresses of the new core peers to the old neighboring cores.
Round 5: The old core broadcasts the new partners of the matchings within $G_v[\cdot, 0]$; this ends the repairing operation with respect to the snapshot's state.

Expansion

When the pancake graph's order is incremented from d to $d+1$, each node v must split into $d+1$ new nodes. Since the grid G_v consists of $d+1$ columns, there is a simple way to perform the expansion on the grid level: every column becomes one new node.

According to Chapter 3.3.1, two neighboring expanded nodes have already been adjacent in P_d (or originate from the same node). Assume that two columns, one in G_v and the other one in G_u, for two expanding adjacent nodes $u, v \in V(P_d)$, become neighbors in P_{d+1}. With the grid as described so far, these two columns have only one connection to each other (one pair of core peers). In order to increase the fault-tolerance the following mechanism is applied: as soon as there are enough peers in the system and there are at least $d+2$ complete rows in each node, adjacent nodes $u, v \in V(P_d)$ start to establish a matching between the columns in G_u and G_v which will become neighbors if the graph is expanded. In order to limit the information that is sent, we establish this matching stepwise, ensuring that it is finished before the node actually has to split. This is done in $d+1$ phases, in phase i for the matching to neighbor $\rho_i(v)$. The idea is that each peer at $G_v[x, y]$ with $y \in [1, d+2]$ sends its IP address to the peer $G_v[y-1, 0]$. Peer $G_v[y-1, 0]$ is then responsible to transfer the y^{th} row to the corresponding peers $G_{\rho_i(v)}[y, 0]$ for $i \in [2, d+2]$. From there, the information is broadcast to $G_{\rho_i(v)}[\cdot, y]$. This mechanism guarantees that between two neighboring columns, at least one connection will be finished, even in the presence of a concurrent adversary. Once the matching is established, it is maintained as long as there are at least $d+2$ rows.

The expansion then works as follows. We consider a node $v = l_1 ... l_d$ with grid G_v. The column $G_v[i, \cdot]$ for $i \in [1, d+1]$ will form the new node $v_{(i)}^{exp} = l_1 ... l_{i-1}(d+1)l_i ... l_d$. Since peers of the i^{th} column $G_v[i, \cdot]$ are completely connected, the expansion can be performed in two rounds: it is straightforward to locally compute the form of the new grids $G_{v_{(i)}^{exp}}$, including cores and inter-connections, and send this information to nodes $\rho_j(v_{(i)}^{exp})$ for $j \in [2, d+1]$.

Round 1: The peers of the i^{th} column $G_v[i, \cdot]$ which will form the new node $v_{(i)}^{exp}$ are completely connected, and each peer in $v_{(i)}^{exp}$ can locally compute the form of $G_{v_{(i)}^{exp}}$. The information about the new core is sent to nodes $\rho_j(v_{(i)}^{exp})$ for $j \in [2, d+1]$, using the connections of the matching.

Round 2: The peers in $v_{(i)}^{exp}$ send the information about the neighboring cores received in Round 1 to their own new core.

Reduction

The reduction of the pancake's order is more elaborate: reducing the order from $d+1$ to d requires $d+1$ grids to merge into one. Additionally, some peers are bound to change nodes (cf Chapter 3.3.1).

Similarly to the notation introduced in Chapter 3.3.1, let $v_{(1)}^{dom} \in V(P_{d+1})$ be the dominator of a cluster that contracts to $v \in V(P_d)$ and let $v_{(i)}^{dom} = \rho_i(v_{(1)}^{dom})$. To reduce the order of the pancake graph, we must exchange the nodes $v_{(i+1)}^{dom}$ with $u_{(i+1)}^{dom}$ for $i \in [2, d]$ where $u = \rho_i(v)$, and then merge the clusters into one node v (cf Chapter 3.3.1).

On the grid level, a constant number of rounds is needed for this order reduction. Basically, the procedure is as follows. First we turn $G_{v_{(i)}^{dom}}$ for

$i \in [1, d+1]$ into a clique and the information about the core of $G_{v_{(1)}^{dom}}$ is sent to $\rho_{i-1}(v_{(i)}^{dom})$ (node exchange, cf Chapter 3.3.1). Now, the new grid of node v will be formed. For this, let again $v_{(i)}^{exp}$ for $i \in [1, d+1]$ be the nodes which will form v after the node exchange, $v_{(1)}^{exp}$ being the dominator. After $v_{(1)}^{exp}$ learned about its new dominated nodes, it sends *all* its peers' addresses to $v_{(i)}^{exp}$ for $i \in [2, d+1]$. With this information, a first version of G_v can be computed, where column i is given by $v_{(i)}^{exp}$. Based on this structure, the final grid can be obtained by a rearrangement.

3.3.5 The System

The n peers in our system are arranged in a simulated pancake topology of order d. The data of the DHT is stored as follows. Let $hash(\cdot)$ be a hash function which, given an identifier ID, outputs a random permutation on some set $[1, N]$, where N is a sufficiently large global integer constant. A data item with identifier ID is stored on the node $v \in V(P_d)$ which is determined by the ordering of the smallest d numbers of $hash(ID)$. A data item is not copied to *all* peers in that node, but only replicated on the core at the bottom row. Recall that this has the advantage that—if we use peers in topmost rows for the peer distribution—unnecessary copying of data can be avoided when peers move between nodes, while we are still able to tolerate the same powerful adversary. Finally, observe that routing is simple in the pancake system: assume that a peer in a node $u = l_1 l_2 ... l_d$ wants to find a data item which hashes to a node $v = \widehat{l_1}\widehat{l_2}...\widehat{l_d}$. The lookup operation proceeds by correcting one "coordinate" at a time, starting at the back: from node $u = l_1 l_2 ... l_d$ the request is forwarded to node $l_d ... l_{j+1} l_1 l_2 ... l_{j-1} \widehat{l_d}$, etc.

We now describe how to assemble the components to form a p2p system resilient to an adversary $\mathcal{A}_{ADV}(\Theta(\log n / \log\log n), \Theta(\log n / \log\log n), 1)$. We permanently run $\mathcal{A}_{IA}$ to estimate the total number of peers in the system and adapt the pancake's order accordingly, $\mathcal{A}_{TD}$ to distribute the peers evenly among the pancake's nodes, and $\mathcal{A}_{GRID}$ to maintain the grid. When the order of the pancake is changed, both $\mathcal{A}_{IA}$ and $\mathcal{A}_{TD}$ are restarted. This is possible because our system guarantees that after a change of the pancake's order, there are sufficiently many rounds without another order change such that the estimations of the total number of peers are up-to-date.

Taking into account that $\mathcal{A}_{IA}$ delivers the estimated number of peers with a delay of $d-1$ phases, and that according to Theorem 3.15, the difference between the total number of peers at any two nodes is bounded by $\mathrm{O}(d)$ if there are $\mathrm{O}(d)$ joins and leaves per time unit, we have the following theorem.

Theorem 3.16. *The pancake p2p system guarantees peer degree and network diameter* $\mathrm{O}(d)$ *in the presence of an adversary who inserts and deletes* $\Theta(d)$ *peers per unit time. Each node always has at least one living core peer and no data is lost. Moreover, it holds that* $d = \Theta(\log n / \log\log n)$*, where* n *is the total number of peers in the system.*

The proof of Theorem 3.16 is similar to the deduction of Theorem 3.7, and is omitted here.

3.4 Concluding Remarks

This chapter presented algorithms to maintain p2p networks under worst-case joins and leaves. Two systems have been described which are optimal in the sense that there cannot exist topologies with a smaller peer degree which are robust to the same amount of churn.

It is often justified to study alternative churn models, e.g., *probabilistic* models [73] where peers join and leave according to a *Poisson process*. However, here we pursue a more conservative approach as this gives stronger guarantees. In addition, more optimistic models do not take attackers or viruses into account which exploit the p2p topology and propagate along the p2p system's links, indeed harming certain parts of the network more severely than others.

We believe that our algorithms can be applied to other topologies as well. All that is needed is a token distribution and information aggregation algorithm on the graph. In particular, it would be interesting to maintain alternative hypercubic structures similar to the ones used in Pastry or *skip graphs* [25, 112], where there is no global dimension change but where the graph can evolve more locally.

So far, our algorithms have not been implemented on a real system. However, it would be interesting to apply some of our findings to our p2p tools Pulsar and Wuala, or to the xPilot game.

Chapter 4

Dynamic Throughput Maximization

While the last chapter was mainly concerned with dynamic joins and leaves of peers, we now study another source of dynamism: the available bandwidth between two peers. The predominant transmission protocol of today's Internet is TCP, and many p2p applications use TCP for their transfers. In the following, we will use the terms "(Internet) host" and "peer" interchangeably to refer to a sender or receiver of a transmission

We will devise and analyze a transmission protocol which seeks to maximize throughput between a sending and a receiving peer, while the Internet congestion varies over time. Again, a worst-case scenario is considered: our analysis assumes that the changes of the available bandwidth are controlled by an adversary.

4.1 Background

Congestion avoidance in the Internet has been studied with zeal for many years. The TCP congestion control mechanism of today's Internet successfully employs a window-based scheme to prevent the network from being overloaded. Thus, the size of the so-called *TCP congestion window* is an approximation of the available network capacity. When a TCP sender suffers a packet loss, it assumes that the network is congested and reduces the window's size. Consequently, the sending rate is cut down, and the Internet hosts collaboratively alleviate the load.

In the past, the transport layer and in particular the congestion problem was first studied empirically, and later embraced by the queuing theory and control theory communities. In order to analyze and compare protocols theoretically, a traffic model is needed. Queuing and control theory researchers have refined their early Poisson traffic models to an astonishing level of detail. However, probabilistic models are intricate to analyze. Probabilistic models that are simple enough to be analytically tractable might never model traffic accurately enough, as the nature of network traffic is self-similar and *bursty*.

In their paper Karp, Koutsoupias, Papadimitriou, and Shenker [126] have proposed to study congestion control from a worst-case perspective instead. Karp et al. model congestion control as an online game between a flow and an adversarial network. In particular, the available bandwidth of the network changes over time and the flow gets only a limited feedback—namely, whether packets have been lost or not—about the currently available bandwidth.

In this chapter, we follow the algorithmic online approach proposed by Karp et al. [126]. We build upon [126] by focusing on the dynamics of congestion; in particular, we integrate a notion of *bursts* happening in a worst-case manner. We first present a new analysis of a model by Karp et al., and then introduce a burst model: instead of considering an adversary who can always change the available bandwidth to the same extent in each round, our adversary can accumulate power in quiet rounds and then change the congestion more abruptly in later rounds. The definition of this adversary is based on *network calculus* concepts.

Network Calculus In A Nutshell

We give a short introduction to those concepts of network calculus which are relevant to our work. Network calculus is a relatively new technique to analyze deterministic queuing systems found in communication networks. For a detailed introduction to network calculus, the reader is referred to the introductory book by Le Boudec and Thiran [139].

In network calculus, there exists the notion of *arrival curves* which provide deterministic limitations to the network traffic sent by sources. Given that the data flows correspond to these limitations, it is possible to make statements about the deterministic behavior of the network (maximal delays, maximal queue lengths, etc.).

Arrival curves are defined as follows. Let R be a data flow, and let $R(t)$ be the total number of bits R has sent until time t. Let α be an increasing function defined for all times $t \geq 0$. We say that R has an arrival curve α if and only if for all $s \leq t$:

$$R(t) - R(s) \leq \alpha(t - s)$$

In other words, the total number of bits sent until time t by flow R may never exceed the bits sent by R until some time s plus $\alpha(t - s)$. We look at a so-called *leaky bucket arrival curve* defined as $\alpha(t) = c_1 t + c_2$ for some non-negative constants c_1, c_2.

Note that such an arrival curve incorporates a limited form of amortization: if flow R only sends a few bits for several rounds, the constraints of earlier rounds get weaker and allow R to send up to c_2 bits at once in later rounds.

4.2 The Optimization Problem

In the Internet, there is no central authority allocating bandwidth to hosts. On the contrary, individual hosts are responsible for setting their sending rate.[1] In this chapter, we consider the problem of regulating the rate of a unicast flow from one peer to another such that the throughput is maximized. The bandwidth available to the flow thereby fluctuates according to the varying requirements for bandwidth of other competing flows. A peer is not provided direct information about the competing demands for bandwidth or the Internet topology, but does receive some limited feedback as to whether the flow is experiencing packet drops. The peer is bound to determine its transmission rate solely on the basis of this information.

We assume that time is divided into infinitely many successive *rounds*. We consider a *worst-case model* where in every round t, an adversary ADV selects the available bandwidth u_t representing the maximum rate at which a sender can transmit without experiencing packet drops. The sending peer runs an algorithm ALG which decides the transmission rate x_t for each round t. The peer receives immediate feedback as to whether packet drops have occurred, i.e., whether $x_t > u_t$. ALG can subsequently choose the rate x_{t+1} depending on the obtained feedback.

We assume a *severe cost model* [126] where a host cannot transmit anything in round t if $x_t > u_t$, but can transmit at a rate x_t if $x_t \leq u_t$. Formally, the *gain* of ALG in round t is defined as follows:

$$gain_{ALG}(x_t, u_t) \;:=\; \begin{cases} x_t & \text{, if } x_t \leq u_t \\ 0 & \text{, otherwise} \end{cases}$$

An optimal offline algorithm OPT knows the sequence $\{u_t\}$ in advance and achieves a gain of

$$gain_{OPT}(x_t, u_t) = u_t$$

in round t. These gains take into consideration two major aspects of transmissions in bandwidth-limited environments: the online algorithm experiences an *opportunity cost* if its sending rate is smaller than the available bandwidth (case $x_t < u_t$), and a *retransmission overhead* if its packets are dropped due to congestion (case $x_t > u_t$).

We are in the realm of *competitive analysis* [52] and define the (strict) *competitive ratio* ρ achieved by ALG as the total amount of data (over all rounds) sent by OPT divided by the total amount of data sent by ALG (cf Definition 4.1).

[1]Usually, this is done automatically by TCP. However, by using the User Datagram Protocol (UDP), selfish programs can try to maximize their own throughput and may have no incentive to reduce congestion collaboratively. Although it is generally believed that routers are configured to give priority to TCP packets [107]—with the consequence that UDP packets are dropped first if the Internet gets congested—at least in theory it is possible to design networking software from scratch that circumvents this restriction by sending UDP packets which look like TCP packets.

Definition 4.1 (ρ-competitive)**.** *We say that an algorithm ALG is* (strictly) ρ-competitive *compared to an optimal offline algorithm OPT if for all input sequences I, it holds that*

$$gain_{OPT}(I) \le \rho \cdot gain_{ALG}(I).$$

The goal of the online algorithm designer is to minimize ρ. Henceforth, we will assume that ALG knows the initial bandwidth, i.e., $x_0 = u_0$.

Observe that an unrestricted adversary could frustrate every online algorithm by always selecting $u_t := x_t - \varepsilon$ for some arbitrary small $\varepsilon > 0$. The natural solution proposed by Karp et al. [126] is to assume that the available bandwidth does not change too drastically over time. In this chapter, we study different ways to restrict the adversary. In Chapter 4.3, we consider the multiplicative model proposed by Karp et al. In Chapter 4.4, we extend this model to allow for changes with *bursts*.

We will call rounds t where the online algorithm successfully transmits its packets without loss *good rounds*, and rounds t where $x_t > u_t$ *bad rounds*, cf Definition 4.2.

Definition 4.2 (Good and Bad Rounds)**.** *A round t where $x_t \le u_t$ is called* good*, a round t where $x_t > u_t$ is called* bad*.*

We defer the description of the different adversaries to the corresponding sections. However, we now define the following class of online algorithms.

Definition 4.3 ($\mathcal{ALG}(G,B)$)**.** *Let $\mathcal{ALG}(G,B)$ be the online algorithm which chooses*

$$x_{t+1} := \begin{cases} G \cdot x_t & , \text{if } x_t \le u_t \\ B \cdot x_t & , \text{otherwise} \end{cases}$$

for some $G \ge 1$ and $B \le 1$. That is, the algorithm $\mathcal{ALG}(G,B)$ increases the rate by a factor G after a good round, and decreases it by a factor B after a bad round.

The sending rate x_{t+1} of an algorithm $\mathcal{ALG}(G,B)$ depends solely on the binary feedback whether its probing rate x_t was larger than the available bandwidth u_t in the previous round or not.

4.3 Multiplicative Adversary

In this section, we look at multiplicative changes of the available bandwidth. We first consider a model where the adversary can increase the bandwidth by a factor of at most $\mu \ge 1$ per round and can decrease it arbitrarily (cf Definition 4.4). Later, we will study a model where also the reduction is constrained by multiplicative factors (cf Definition 4.5).

The adversary $\mathcal{ADV}_{mult}$ proposed by Karp et al. is defined as follows.

Definition 4.4 ($\mathcal{ADV}_{mult}$). *$\mathcal{ADV}_{mult}$ chooses the new bandwidth u_{t+1} in the interval $[0, u_t \cdot \mu]$, i.e.,*

$$\mathcal{ADV}_{mult} : u_{t+1} \in [0, u_t \cdot \mu],$$

for some given $\mu \geq 1$.

First, we restate the lower bound given in [126].

Theorem 4.1. *[126] Against $\mathcal{ADV}_{mult}$, no online algorithm can achieve a competitive ratio smaller than μ.*

Proof. Consider the following adversary ADV: in every round t, he chooses

$$u_t := \begin{cases} \mu & \text{, if } x_t \leq 1 \\ 1 & \text{, otherwise} \end{cases}$$

Thus, whenever an online algorithm ALG sends at a rate larger than one, all its packets are dropped because of congestion. On the other hand, if ALG transmits at a rate of 1 or less, the rate of OPT is at least a factor μ larger. Moreover, since ADV changes the available bandwidth at most by a factor of μ per round, he is indeed of type $\mathcal{ADV}_{mult}$. $\square$

In [126], it is shown that the algorithm $\mathcal{ALG}(\mu, \frac{\sqrt{\mu}}{\sqrt{\mu}+\sqrt{\mu-1}})$ yields a competitive ratio of

$$\rho = (\sqrt{\mu} + \sqrt{\mu - 1})^2$$

against $\mathcal{ADV}_{mult}$. However, [126] uses a different definition for the competitive ratio which allows for additive constants. By our strict definition (cf Definition 4.1), the ratio can be much larger. To see this, assume an adversary who reduces the available bandwidth in every round by a factor slightly larger than $\frac{\sqrt{\mu}+\sqrt{\mu-1}}{\sqrt{\mu}}$. In this case, $\mathcal{ALG}(\mu, \frac{\sqrt{\mu}}{\sqrt{\mu}+\sqrt{\mu-1}})$ is only successful in the first round, and hence $gain_{ALG} = u_0$, while

$$gain_{OPT} \approx u_0 \cdot \sum_{i=0}^{\infty} (\frac{\sqrt{\mu}}{\sqrt{\mu} + \sqrt{\mu - 1}})^i.$$

Therefore, the (strict) competitive ratio is

$$\rho = \frac{gain_{OPT}}{gain_{ALG}} \approx \frac{\sqrt{\mu} + \sqrt{\mu - 1}}{\sqrt{\mu - 1}}.$$

For small μ, ρ is large (for instance $\rho > 100$ if $\mu = 1.0001$).

In the following, we will give a simple proof that the algorithm $\mathcal{ALG}(\mu, 1/2)$ has a strict competitive ratio 4μ. According to Theorem 4.1, this is asymptotically optimal.

Theorem 4.2. *$\mathcal{ALG}(\mu, 1/2)$ is 4μ-competitive against $\mathcal{ADV}_{mult}$.*

Proof. First, we show by induction that in every good round t, $u_t \leq 2\mu x_t$. For $t = 0$, $u_0 = x_0$ and the claim holds. For the induction step, consider the round $t-1$ before the good round t. There are two possibilities: either round $t-1$ has been bad ($x_{t-1} > u_{t-1}$) or good ($x_{t-1} \leq u_{t-1}$). If round $t-1$ has been bad, we have $x_t = x_{t-1}/2$ and $u_t \leq u_{t-1}\mu < x_{t-1}\mu = 2\mu x_t$, hence $u_t/x_t < 2\mu$, and the claim holds. If on the other hand round $t-1$ was good, the algorithm increases the bandwidth at least as much as the adversary. Together with the induction hypothesis, the claim also follows in this case.

Having studied the gain in good rounds, we now consider bad rounds. We show that in the bad rounds following a good round t, the adversary can increase his gain at most by $2\mu x_t$. So let t be the good round preceding a sequence of bad rounds, i.e., $x_t \leq u_t$, $x_{t+1} > u_{t+1}$, $x_{t+2} > u_{t+2}$, etc. We know that $x_{t+1} = \mu x_t$, so—because it is a bad round—u_{t+1} must be smaller than μx_t. Furthermore, we have $x_{t+2} = x_{t+1}/2 = \mu x_t/2$ and hence $u_{t+2} < \mu x_t/2$, $x_{t+3} = \mu x_t/4$ and hence $u_{t+3} < \mu x_t/8$, and so on. By a geometric series argument, the gain of the adversary in the bad rounds is upper bounded by $2\mu x_t$.

Therefore,

$$\begin{aligned}
\rho &= \frac{gain_{OPT}(good) + gain_{OPT}(bad)}{gain_{ALG}(good)} \\
&< \frac{2\mu \cdot gain_{ALG}(good) + 2\mu \cdot gain_{ALG}(good)}{gain_{ALG}(good)} \\
&= 4\mu.
\end{aligned}$$

□

To conclude this section, we give another type of proof to show that the algorithm $\mathcal{ALG}(\mu, 1/\mu^3)$ has a good competitive ratio for small μ. For our analysis, we assume a slightly more restricted adversary $\mathcal{ADV}^*_{mult}$ (cf Definition 4.5).

Definition 4.5 ($\mathcal{ADV}^*_{mult}$). *$\mathcal{ADV}^*_{mult}$ chooses the new bandwidth u_{t+1} from the interval $[u_t/\mu, u_t \cdot \mu]$, i.e.,*

$$\mathcal{ADV}^*_{mult} : u_{t+1} \in [u_t/\mu, u_t \cdot \mu].$$

Theorem 4.3. *$\mathcal{ALG}(\mu, 1/\mu^3)$ is $(\mu^4 + \mu)$-competitive against $\mathcal{ADV}^*_{mult}$.*

Proof. The fact that ALG reduces its rate by a factor μ^3 after a bad round implies that the next round is always good: assume, for the sake of contradiction, that round $t+1$ is the first bad round following another *bad* round t, which—by the induction hypothesis—follows a good round $t-1$. Hence, $x_{t-1} \leq u_{t-1}$. Moreover, observe that $u_{t+1} \geq u_t/\mu \geq u_{t-1}/\mu^2$, but on the other hand, $x_{t+1} = x_t/\mu^3 = \mu x_{t-1}/\mu^3 = x_{t-1}/\mu^2$. Therefore, $x_{t+1} \leq u_{t+1}$. Contradiction!

We now first analyze the gain of a good round t and show that $u_t < \mu^4 x_t$. There are two cases: either round $t-1$ has also been good, or not. If it has been a good round, then round t is at least as competitive as round $t-1$ because $x_t = \mu x_{t-1}$. If on the other hand round $t-1$ has not been good, we have $u_{t-1} < x_{t-1}$, $x_t = x_{t-1}/\mu^3$ and $u_t \leq \mu u_{t-1}$. Therefore, $x_t = x_{t-1}/\mu^3 > u_{t-1}/\mu^3 \geq u_t/\mu^4$, and the claim follows.

Next, we study the gains in a bad round t. In this case, it holds that $u_t < \mu x_{t-1}$: since $x_{t-1} \leq u_{t-1}$, $x_t = \mu x_{t-1}$ and $u_t < x_t$, and hence $u_t < \mu x_{t-1}$.

Therefore,

$$\begin{aligned}\rho &= \frac{gain_{OPT}(good) + gain_{OPT}(bad)}{gain_{ALG}(good)} \\ &< \frac{\mu^4 \cdot gain_{ALG}(good) + \mu \cdot gain_{ALG}(good)}{gain_{ALG}(good)} = \mu^4 + \mu.\end{aligned}$$

□

Since $\mathcal{ADV}^*_{mult}$ is a special case of $\mathcal{ADV}_{mult}$, Theorem 4.2 also applies for $\mathcal{ADV}^*_{mult}$. Hence, it is possible to run $\mathcal{ALG}(\mu, 1/\mu^3)$ against $\mathcal{ADV}^*_{mult}$ if μ is small, and $\mathcal{ALG}(\mu, 1/2)$ otherwise, which yields the following corollary.

Corollary 4.4. *There is a deterministic algorithm which is* $\min\{\mu^4 + \mu, 4\mu\}$-*competitive against* $\mathcal{ADV}^*_{mult}$.

4.4 Network Calculus Adversary

4.4.1 Description of $\mathcal{ADV}_{nc}$

We now introduce the adversary $\mathcal{ADV}_{nc}$ whose definition is based on network calculus concepts. We will extend the model introduced in Chapter 4.3 by a form of limited amortization which allows for more drastic bandwidth changes after times of quiescence.

$\mathcal{ADV}_{nc}$ has two parameters: a *rate* $\mu \geq 1$ and *maximum burst factor* $B \geq 1$. In every round, the available bandwidth u_t varies according to these parameters in a multiplicative manner. More formally, $\mathcal{ADV}_{nc}$ selects the new bandwidth u_{t+1} from the interval

$$\mathcal{ADV}_{nc} : u_{t+1} \in [\frac{u_t}{\beta_t \mu}, u_t \cdot \beta_t \cdot \mu],$$

that is, the available bandwidth can change by a factor of at most $\beta_t \mu$. Here, β_t is the *burst factor at time t*. This factor is explained next.

On average, the available bandwidth can change by a factor μ per round. However, there may be times of only small changes, after which the available bandwidth can change by factors larger than μ. This is modeled with the burst factor β_t: at the beginning, β_t equals B, i.e., $\beta_0 = B$. For $t > 0$, the burst factor β_t is computed depending on β_{t-1} and the actual bandwidth change c_{t-1} that has happened in round $t-1$. More precisely,

$$\beta_t = \min\{B, \beta_{t-1} \frac{\mu}{c_{t-1}}\},$$

where

$$c_t := \begin{cases} \frac{u_{t+1}}{u_t} & \text{, if } u_{t+1} > u_t \\ \frac{u_t}{u_{t+1}} & \text{, otherwise} \end{cases}$$

This means that if the congestion has changed by a factor less than μ in round t, i.e., $c_t < \mu$, the burst factor *increases* by a factor μ/c_t, and hence the available bandwidth can change more in the next round—and vice versa if $c_t > \mu$.

Hence, the adversary can save power for forthcoming rounds. However, this amortization is limited as β_t is never larger than B for all rounds t. Also note that $\forall t : \beta_t \geq 1$, as $c_t \leq \mu\beta_t$ by the definition of $\mathcal{ADV}_{nc}$.

4.4.2 Analysis

At first sight, it seems that $\mathcal{ADV}_{nc}$ has roughly the same power as $\mathcal{ADV}^*_{mult}$: in order to change the available bandwidth by a factor larger than μ, $\mathcal{ADV}_{nc}$ must have changed the bandwidth by a factor smaller than μ in previous rounds.[2] However, as we will show in the following, an online algorithm cannot exploit these quiet rounds sufficiently, and the competitive ratio can grow for larger B.

Theorem 4.5. *The competitive ratio is at least $\Omega\Big(\mu\sqrt{B}/\log B\Big)$ against $\mathcal{ADV}_{nc}$.*

Proof. Consider the following adversary ADV. ADV selects $u_t = 1$ whenever the burst factor β_t is not maximal in a round t, i.e., if $\beta_t < B$. If $\beta_t = B$, ADV continues choosing $u_t = 1$ until $x_t \leq 1$ for the first time. Then, if $x_t \leq 1$ and $\beta_t = B$, he selects $u_t = \mu\sqrt{B}$ but immediately resets the available bandwidth to $u_{t+1} = 1$ in the next round. Therefore, no online algorithm can ever transmit at a rate larger than 1. Since ADV must be of type $\mathcal{ADV}_{nc}$, he can do this trick at most every $\lceil \log B/\log\mu \rceil$ rounds: after these two bursts (from 1 to $\mu\sqrt{B}$ and from $\mu\sqrt{B}$ back to 1), the burst factor becomes 1, and it takes $\lceil \log B/\log\mu \rceil$ rounds to increase it again to B: $\mu^i \geq B \Leftrightarrow i \geq \log B/\log\mu$.

Let us call the time period between two rounds where ADV raises the bandwidth from 1 to $\mu\sqrt{B}$ a *phase*. In every phase, ALG has a gain of at most $gain_{ALG} \leq 2 + \lceil \log B/\log\mu \rceil$. On the other hand, the optimal algorithm's gain is at least $gain_{OPT} \geq 1 + \lceil \log B/\log\mu \rceil + \mu\sqrt{B}$. Hence,

$$\rho \;=\; \frac{gain_{OPT}}{gain_{ALG}} \geq \frac{1 + \lceil \log B/\log\mu \rceil + \mu\sqrt{B}}{2 + \lceil \log B/\log\mu \rceil} \in \Omega\left(\mu\frac{\sqrt{B}}{\log B}\right).$$

□

[2]Except for the first rounds of course, where a burst B comes "for free". However, as mentioned in Chapter 4.2, we consider infinite games only.

Note that the lower bound given in Theorem 4.5 even holds for online algorithms which get perfect (instead of only binary) feedback about the bandwidth of the previous round.

Although we were not able to find an algorithm which yields a tight upper bound, it can be shown that $\mathcal{ALG}(\mu\sqrt[3]{B}, 1/2)$ comes close to the bound of Theorem 4.5.

Theorem 4.6. *The competitive ratio of $\mathcal{ALG}(\mu\sqrt[3]{B}, 1/2)$ is $O\left(\mu^{3/2}B^{2/3}\right)$ against $\mathcal{ADV}_{nc}$.*

Proof. We apply the proof technique of Chapter 4.3. First, we analyze the missed gain in bad rounds:

$$\begin{aligned} gain_{OPT}(bad) &\leq \sum_{i=0}^{\infty}\left(\frac{1}{2}\right)^i \cdot \mu\sqrt[3]{B} \cdot gain_{ALG}(good) \\ &\leq 2\mu\sqrt[3]{B} \cdot gain_{ALG}(good) \\ &\in O\left(\mu\sqrt[3]{B}\right) \cdot gain_{ALG}(good). \end{aligned}$$

Next, the good rounds are tackled. Let t be the last bad round before a good round $t+1$. Hence, $x_t > u_t$, $x_{t+1} = x_t/2 \leq u_{t+1}$, and $x_{t+2} = \mu\sqrt[3]{B}x_t/2$.

There are two cases: either round $t+2$ is also good, or not. If round $t+2$ is good, $u_{t+2} \leq \mu^2 B x_t$. We have

$$\rho \leq \frac{u_{t+1} + u_{t+2}}{x_{t+1} + x_{t+2}} \leq \frac{\mu B + \mu^2 B}{1/2 + \mu\sqrt[3]{B}/2} \in O\left(\mu B^{2/3}\right).$$

More good rounds would reduce this ratio, because ALG grows faster than ADV.

If round $t+2$ is not good, it holds that $x_t > u_t$ and $x_{t+2} = \mu\sqrt[3]{B}x_t/2 > u_{t+2}$. Now observe that $u_{t+1} < \mu^{3/2}B^{2/3}x_t$. Assume, for the sake of contradiction, that $u_{t+1} \geq \mu^{3/2}B^{2/3}x_t$. Then the burst factor in round $t+1$ is at most $\beta_{t+1} \leq \sqrt[3]{B}/\sqrt{\mu}$, and thus

$$u_{t+2} \geq \frac{u_{t+1}}{\mu\beta_{t+1}} \geq \frac{\mu^{3/2}B^{2/3} \cdot \sqrt{\mu}}{\sqrt[3]{B} \cdot \mu}x_t = \mu\sqrt[3]{B}x_t > x_{t+2}.$$

Contradiction. Hence,

$$\rho \leq \frac{u_{t+1}}{x_{t+1}} \leq \frac{\mu^{3/2}B^{2/3}x_t}{x_t/2} \in O\left(\mu^{3/2}B^{2/3}\right).$$

Thus,

$$\begin{aligned}
\rho &= \frac{gain_{OPT}(good) + gain_{OPT}(bad)}{gain_{ALG}(good)} \\
&\leq \frac{O\left(\mu^{3/2}B^{2/3}\right) \cdot gain_{ALG}(good) + O\left(\mu\sqrt[3]{B}\right) \cdot gain_{ALG}(good)}{gain_{ALG}(good)} \\
&\in O\left(\mu^{3/2}B^{2/3}\right).
\end{aligned}$$

□

4.5 Concluding Remarks

We have given an analysis of different transfer protocols which seek to selfishly maximize throughput between a sender and a receiver. In doing so, we made the realistic assumption that peers do not know the bandwidth available to them in the future, and showed that it is still possible to be worst-case competitive against an offline algorithm which has perfect knowledge of the system's state. In addition, we introduced a new way to model dynamic phenomena which are bursty in nature.

Many questions remain open. An obvious question is whether the lower bound and the upper bound can be made tight. Another challenge is the design of *randomized* online algorithms. In fact, Arora and Brinkman [22] have addressed this problem for the multiplicative adversary $\mathcal{ADV}_{mult}$ and presented an algorithm with competitive ratio $O(\log \mu)$. By using *Yao's minimax principle* [52], it can be shown that this is asymptotically optimal. However, the authors assume a weak oblivious adversary: their scheme uses randomization only in the first round, while all later rounds are deterministic. But the adversary is not allowed to be adaptive even in these deterministic rounds. The case of a stronger adversary is still an open problem. It is straight-forward to extend the algorithm by Arora and Brinkman for $\mathcal{ADV}_{nc}$. However, also here, it would be interesting to study a more powerful adversary who can be adaptive in deterministic rounds.

Chapter 5

Related Work

Part I of this thesis has studied p2p networks where the topology and the bandwidth is dynamically changing. This chapter compares our work to related literature. We first review papers on churn, and subsequently discuss literature in the area of congestion control.

On the one hand, p2p systems are potentially more resilient compared to alternative systems as they do not rely on central servers which constitute a single point of failure. On the other hand, in p2p networks, there are frequent joins and leaves, and hence failures are more common. When designing robust p2p applications such as online storage systems like Wuala, it is beneficial to have an idea of the dynamic nature of transient and permanent failures the system will be confronted with in order to optimize system performance and to include appropriate levels of redundancy.

Unfortunately, measuring the dynamics of existing p2p systems is not always simple [108]; moreover, in this emerging area where the space of possible applications is still not well understood, it is difficult to generalize a given measurement to entire application classes [232]. Nevertheless, there have been several measurement studies of p2p churn. Already in 2002, Saroiu et al. [210] reported on the dynamic nature of Napster and Gnutella, and Sen et al. [214] analyzed flow-level data of a large ISP to estimate churn. The availability of the *Overnet* system has been studied in [47] by probing crawled hosts; the authors showed that previous works had underestimated host availability due to a methodological limitation. Qiao et al. [195] reported a similar level of dynamics in Gnutella and Overnet. Recently, based on *Maze* [245] measurements—one of the largest p2p systems over the China education and research network—Tian and Dai [232] pointed out that our understanding of peer dynamics is far from adequate, and that previous crawler based measurements can only yield inaccurate results. Moreover, their data reveals that newly registered peers generally have a higher turnover rate than elder ones. This observation has been confirmed by measurements on the Kad network [223, 227]: peer arrivals do not follow a Poisson distribution and session times are typically not distributed exponentially. It has also been discovered that the levels of churn vary depending on the application. For instance, Guha et al. [106] showed that Skype has a higher host availability than other p2p systems.

The effects of churn have been reported in several papers. The price of churn manifests itself in terms of dropped messages, data inconsistency, increased user-experienced latencies or increased bandwidth use [103, 144, 200]. It has been shown that even in stable and managed infrastructures like *PlanetLab*[1], there can be a significant rate of node failure due to nodes becoming extremely slow suddenly and unpredictably [199].

Scant attention is devoted to *theoretic* questions on the dynamics of p2p networks. Protocols such as Pastry [207] and CAN [198] allow for unexpected failures, and it is shown that they remain well-structured after failures occur. However, for this analysis, an ideal initial state is assumed, and it is not shown how a system can return to the initial state after the membership changes [147]. Moreover, maintenance costs can be unnecessarily high. For example, in CAN, a background stabilization process is used which introduces a constant overhead [1].

Two parameters play an important role in the robustness analysis of graphs. The so-called *expansion* of a topology measures the impact peer failures can have on disconnecting intact peers from the rest of the network (e.g., [42]). Interestingly, the expansion of a graph cannot be used to predict how well a network can sustain *random faults*. The *span* parameter [42] has recently been introduced as a good alternative for stochastic analyses.

In the last few years, Awerbuch and Scheideler have proposed several interesting algorithms to render today's peer-to-peer systems more robust. In [34], searchable concurrent data structures are studied where data elements can be stored on a dynamic set of nodes, e.g., in a peer-to-peer network. Their *Hyperring* data structure has degree $O(\log n)$ and requires $O(\log^3 n)$ work for insert and delete operations; search time and congestion is bounded by $O(\log n)$ with high probability, which improves on alternative structures, e.g., the deterministic *Skipnet* by Harvey and Munro [111]. The lack of admission control in p2p networks has triggered researchers to investigate robust join mechanisms [93]. In [212], Scheideler considers a game between a join algorithm and an adversary. The join algorithm includes both n "good" and ϵn (for some fixed constant $\epsilon < 1$) "bad" peers in a ring structure, and seeks to ensure that there is no sequence of size $\Theta(\log n)$ of consecutive peers in which at least half of the peers are malicious. The *k-rotation* strategy is presented which wins with high probability as long as $\epsilon < 1/3$. The scalable solution to join-leave attacks presented in [36] requires a robust random number generator described in [35].

A promising class of novel p2p architectures is due to Naor and Wieder [169]. The authors describe a dynamic decomposition of a 1-dimensional continuous ID space into cells corresponding to processors, and show how to approximate de Bruijn graphs or hypercubes. In particular, their *distance halving* DHT yields an optimal tradeoff in the sense that peer degree d implies a network di-

[1]PlanetLab is a wide-area service deployment testbed spread across North America, Europe, Asia and the South Pacific. See http://www.planet-lab.org/.

ameter of $O(\log_d n)$. Besides achieving a low congestion of $O(\log_d n/n)$, their system is also provably robust against random faults. Finally, a construction for dynamic *expanders* is proposed which is based on the tessellation of the plane.

Liben-Nowell, Balakrishnan, and Karger [147] have analyzed the evolution of p2p systems in the face of concurrent joins and unexpected departures. They give a lower bound for the rate at which peers in the Chord system must participate to maintain the system's distributed state. For instance, they prove that if churn can be described by a Poisson distribution, a peer which receives fewer than k notifications per *half-life* will be disconnected from the network with probability at least $(1 - 1/(e-1))^k$. The half-life time period is defined as the time which elapses in a network of n live peers before n additional peers arrive, or before half of the peers depart. Their result implies that a successor list of length $\Theta(\log n)$ per peer is sufficient to ensure that a graph stays connected with high probability, as long as $\Omega(\log n)$ rounds pass before $n/2$ peers fail. It is also shown in [147] that a modified version of Chord is within a logarithmic factor of the optimal rate. The authors assume that the half-life is known, and the question of how to learn the correct maintenance rate of the behavior of neighbors is left for future research. In [181], a model similar to the one proposed in [147] is considered; however, in this paper, it is assumed that a central server can direct joins to specific locations in the network, and can update old peers' neighbors when a peer leaves. In this scenario, they are able to maintain connectivity with a constant amount of space per peer.

A recent paper by Godfrey et al. [103] has studied—using real-world traces—how to manage churn through the judicious selection of nodes are likely to remain up for a long time. By comparing different algorithms, the authors found that a uniform-random replacement strategy performs quite well. In [164], an analytical framework based on percolation theory is proposed to assess the robustness of p2p systems. Besides a probabilistic churn model, they also consider a targeted attack where nodes having high degrees are progressively removed.

Resilience to *worst-case failures* has been studied by Fiat, Saia et al. in [92, 208]. The authors introduce a system where a $(1-\varepsilon)$-fraction of peers and data survives the adversarial removal of up to half of all nodes with high probability. However, in contrast to our work, the failure model is static. Moreover, the whole structure is designed with a rough a-priori knowledge of the total number of participants, and has to be rebuilt from scratch if the number of peers changes by a constant factor. Abraham et al. [1] address scalability and resilience to worst-case joins and leaves, and propose a generic overlay emulation approach for graph families such as *hypercubes*, *butterflies*, or *de Bruijn* networks. They focus on maintaining a balanced network rather than on fault-tolerance in the presence of concurrent faults. In contrast to our system, whenever a join or leave takes place, the network is given some time to adapt. To the best of our knowledge, the first paper treating concurrent worst-case joins and leaves is by Li et al. [145]. In contrast to our work, Li et al. consider a completely asynchronous model where messages can be arbitrarily delayed. The stronger communication model is compensated by a weaker failure model where leaving peers execute an "exit" protocol which does not allow for sudden crashes.

Finally, observe that there is also work on robust *supervised* overlay networks [135], where the p2p topology is managed by one or more special machines. In these systems, the goal is to minimize information and computational overhead at the central servers while ensuring good network degree, diameter, and expansion. While such an approach can be interesting in centralized environments such as grid computing systems or games, such a solution does not scale and the network entry point constitutes a single point of failure.

So far, neither the hypercube nor the pancake system have been implemented. However, based on the ideas presented in Chapter 3, we have developed *eQuus* [155], a DHT which connects peers in such a way that the peer degree and network diameter is always bounded by $O(\log n)$ with high probability, where n is the total number of peers in the network. eQuus gives less guarantees regarding the worst-case efficiency of the topology under churn. However, in contrast to the work presented in Chapter 3, it is *locality-aware* in the sense that communication takes place along routes whose latencies are close to optimal. It is easy to build applications on top of the eQuus DHT, for instance, a *chat tool* [119].

Note that there also exists literature on algorithms for graphs with *dynamic edges* [211] rather than with dynamic nodes. The availability of communication links in wireless networks can change over time due to interference or mobility of the nodes, which renders basic operations such as routing more difficult [97, 176]. Already in 1988 [39], a *dynamic synchronizer* has been proposed which shows that dynamic asynchronous networks are as fast as static synchronous ones. The so-called *temporal networks* [46, 130] describe situations arising in epidemiology or scheduled transportation networks such as airline travel, where links are only available in certain time periods. Each edge in a graph is annotated with a time label specifying the time at which its endpoint communicated. Given these labels, the goal is, e.g., to find the maximum disjoint time-respecting paths between pairs of nodes. It has been shown in [130] that the classic *Menger' Theorem* cannot be applied in this context anymore.

Finally, the pancake graph used in Chapter 3 and the unsolved problem of computing its diameter was introduced in [83]. In terms of the group-theoretic model for network topologies introduced by Akers and Krishnamurthy [11], the pancake is an instance of a hierarchical *Cayley graph* [20]. We are not aware of any literature on dynamic token distribution or information aggregation on pancake graphs.

Chapter 4 has studied the performance of different transfer protocols in a network where the available bandwidth changes in a worst-case manner. The most prominent transfer protocol is TCP which lies at the heart of today's Internet. Many aspects of TCP have been subject to active research, and several modifications of TCP have been proposed over the years. For instance, protocols such as *STCP*, *Fast TCP*, or *XCP* are used for specialized applications with long-living flows, and also *Stanford's clean slate project*^{2} is devising a new congestion control algorithm for the "future Internet", called the *Rate Control Protocol* (RCP). For a reference on TCP, the reader is referred to [225].

Chapter 4 analyzes congestion control from a *worst-case perspective* using competitive analysis. Whereas all other layers have received quite a lot of attention in the past (e.g., cf [14] for the link layer, and [140] for the network layer), there is less algorithmic networking research about the transport layer. Some notable exceptions are for instance *adversarial queuing theory* [53, 157], *mechanism design* [89], or the study of the *TCP ACK problem* [124].

We build upon the model proposed by Karp et al. [126] who define several optimization problems related to congestion control. The authors investigate the issue of regulating the rate of a single unicast flow when the bandwidth available to it is unknown and changes over time. Many results are derived. For a *static case* where the available bandwidth is drawn from $\{1, ..., n\}$, an upper bound of $O(n \log \log n)$ is given, together with a lower bound of $\Omega(n \log \log n / \log \log \log n)$. For the dynamic case where the bandwidth is chosen *from a fixed interval* an interval $[a, b]$, the optimal deterministic ratio is a/b, and the optimal randomized ratio against an oblivious adversary is $1 + \ln(a/b)$. If the the changes are bounded multiplicatively from $[u_t/\mu, \mu u_t]$, the bound is at most $4\mu - 2$, and no deterministic algorithm can be better than μ. Finally, for an additive scenario where the new ratio is chosen from an interval $[u_t - \alpha, u_t + \alpha]$, the deterministic ratio is between $1 + \alpha/\beta$ and $4 + \alpha/\beta$, where β is the absolute lower bound of the bandwidth. Chapter 4 extends [126] in two respects: first, we give a new analysis of a model where the bandwidth changes multiplicatively; our analysis is simpler and gives *strict* competitive bounds. Second, we enhance their model with *bursts*.

The work by Karp et al. has already had an interesting follow-up by Arora and Brinkman [22] who study *randomized* algorithms for a dynamically changing congestion. In particular, they propose an asymptotically optimal randomized online algorithm against an adversary who can change the congestion by a constant factor in every round. Unfortunately, they assume a fairly weak oblivious adversary: their algorithm uses randomization only in the first round, while the sending rate of all other rounds is computed deterministically. The adversary however is not allowed to be adaptive in these deterministic rounds.

The idea that an adversary can accumulate power over time has already appeared in the area of packet routing and is related to the adversarial queuing theory by Borodin et al. [53]. The problem considered there is as follows: given a packet switched network and an adversary who continuously injects packets that have to be routed from a source to a destination node, the goal is to determine the delivery times and the amount of buffer space needed at the nodes. This model offers many interesting research questions. For instance, Lotker, Patt-Shamir and Rosén [157] have shown that the first-in-first-out (FIFO) scheduling protocol can be instable at any injection rate larger than $1/2$ and that it is always stable if the rate is less then $1/d$, where

^{2}See http://cleanslate.stanford.edu/.

d is the length of the longest route used by any packet. Moreover, every work-conserving scheduling algorithm is stable if the rate is smaller than $1/(d+1)$.

In the paper by Aiello et al. [8], the adversary is allowed to inject any sequence of packets into the network, as long as in any w consecutive rounds, the total load created by the paths associated with the packets inserted in this time period is at most wr on any edge, for some $w \geq 1, r \leq 1$. The adversary studied by Andrews et al. [17] is similar to our adversary. Given two parameters $b \geq 1, r \leq 1$, for any $T \geq 1$ consecutive time steps, the adversary may inject as many packets as he wants, as long as the total load created by the paths associated with these packets is at most $Tr + b$ on any edge. These two adversary models have been compared by Rosén in [202]. A contribution of Chapter 4 is to introduce a modified version of the adversary in [17] on the *transport layer*.

Finally, many questions related to congestion control in the Internet are also related to game theory (cf Part II of this thesis): what happens to the system performance if all participants in the network seek to selfishly maximize their throughput? Researchers have also studied the question of which game the TCP/IP congestion control mechanism is a *Nash equilibrium* [128, 183].

Chapter 6

Conclusion

The dynamic composition of resources is one of the main algorithmic challenges in peer-to-peer computing. With the advent of p2p computing, for the first time, the power of a system was actually *based on dynamics*: dynamics became a first-class citizen which had to be taken into account by all components of the distributed system. In contrast to traditional systems which could treat failures as undesirable but rare events, membership changes in peer-to-peer systems are an inherent ingredient of the paradigm. In this paradigm, it is out of question to replace a defective machine by a new one at the cost of a short period of system unavailability. Rather, a p2p system has to accept—or even make use of!—the presence of faulty participants and provide a seamless operation despite ongoing changes. We believe that especially node dynamics is one of the few genuine research challenges which are new in p2p computing and for which there does not exist much literature in other areas of computer science research. Indeed, researchers have started applying techniques from other disciplines such as physics and control theory to gain deeper insights into these aspects.

Part I of this thesis has studied different aspects of p2p dynamics. Chapter 3 presented algorithms to maintain p2p networks under continuous churn. In order to achieve strong guarantees, a conservative scenario was assumed where peers join and leave in a worst-case fashion. A crucial direction for future research is to incorporate a *self-stabilizing mechanism* [30, 33, 77, 79] (the idea of self-stabilization in distributed computing first appeared in the classic paper by E. W. Dijkstra in 1974 who considered self-stabilization in a token ring [77]) which allows the topology to quickly recover from *any* state. Observe that, so far, we do not handle the case where the number of simultaneously crashing peers exceeds the limitations given by our adversary. For example, it would be desirable that if all peers in a node have left, some surviving peers could simulate that node and try to reconstruct it, or to initiate a dimension reduction. Unfortunately, today, there is only few work that rigorously studies self-stabilization issues for overlay networks, and the few known results often provide fast self-stabilization only from a certain *subset* of degraded states that appear to be most relevant. An interesting publication in this context is by Aspnes et al. [19] who present asynchronous algorithms for rapidly constructing overlay networks from a weakly-connected initial pointer graph. Such fast algorithms can facilitate the construction of hypercubic topologies from arbitrary states. Onus et al. [178] investigate this problem from a self-stabilization perspective.

It would also be interesting to investigate lower bounds for the costs of maintaining peer-to-peer networks. Note that as a p2p system is intended to run continuously, maintenance protocols cannot be evaluated in terms of running time, or total network bandwidth which is infinite if the network persists indefinitely. A better measure is the *rate* at which peers have to expend resources in order to maintain the system. For instance: how much bandwidth is used per join or leave to ensure connectivity, or to maintain a hypercubic network with a logarithmic diameter? A stimulating paper in this regard is [147]. For instance, a lower bound for the consumed network bandwidth is derived in a Poisson model. It could be worthwhile to generalize this approach to other distributions and objective functions, and compare the obtained results to the rates of the algorithms proposed in Chapter 3. Observe that the focus of Chapter 3 was on the feasibility of maintaining overlay structures under churn, and although there are mechanisms which seek to reduce unnecessary data transmissions (e.g., the use of peripheral peers), the message complexity can still be improved.

Many alternative adversarial models of dynamism remain to be studied as well. One avenue for future research is to investigate adversaries who can accumulate power over certain time periods, e.g., to crash a larger number of peers after a quiet period. A way to model such an adversary has been introduced in Chapter 4 in a scenario incorporating another source of dynamism in p2p networks: dynamic changes of the available bandwidth. In contrast to node churn, dynamic bandwidth changes are not specific to peer-to-peer computing, but appear in many application domains on the Internet. Although we do not claim that our model perfectly reflects reality, we believe that it is an interesting simplification which can be useful to gain deeper insights into such behavior.

In this early stage of research, simple models have to be found which are analytically tractable but which at the same time incorporate important aspects of realistic environments. Armed with these results, the goal is to generalize the models. One interesting question in this connection is how dynamic processes with periodic quiet time intervals are related to processes with continuous changes. For example, it seems that there are circumstances in which the results, obtained in a simple model where the adversary and the (maintenance) algorithm take turns, are applicable to settings where the adversary and the repairing algorithm run *concurrently*.

Peer-to-peer dynamics is not only an important but also a challenging and—in our opinion—exciting area of research. The topic being still quite young, the first results are only emerging and many questions remain wide open. Finding answers to these questions can be rewarding not only for computer scientists and p2p users, but also for researchers studying alternative dynamic systems, for instance neurologists or economists. We consider our contributions as a further step to obtain a better understanding of how future architectures can cope with transient participants in order to offer a high availability and correctness.

Part II

Cooperation

Chapter 7

Introduction

The power of peer-to-peer computing arises from the collaboration of the system's constituent parts, the peers. If all the participating peers contribute some of their resources—for instance bandwidth, memory, or CPU cycles—highly scalable decentralized systems can be built which outperform existing server-based architectures. However, in reality, peers may act selfishly and strive for maximizing their own utility by benefiting from the system without contributing much themselves. Hence the performance—and thus its success in practice!—of a p2p system crucially depends on its capability of dealing with selfishness.

A well-known mechanism designed to cope with the free-riding problem is the *tit-for-tat policy*. It is also employed by the file-distribution tool *BitTorrent* which is currently one of the most popular applications on the Internet and which is widely believed to effectively enforce cooperation among peers. In Chapter 8, a BitTorrent case study is conducted which shows that in contrast to common belief, BitTorrent can still be cheated by selfish users. We present the BitThief client which downloads entire files without uploading any real data at all. This is still true in scenarios where there are no so-called seeders. Sharing communities are especially appealing to BitThief.

Motivated by our findings of the BitTorrent case study, we are interested in the impact of selfish behavior in p2p systems. Game theory provides tools to quantify such effects. In Chapter 9, we argue that selfish behavior in peer-to-peer networks has implications beyond the peer's unwillingness to contribute bandwidth or memory. For example, in unstructured p2p systems—still a popular p2p architecture in today's Internet—a peer can often select to which and to how many other peers in the network it wants to connect. With a clever choice of neighbors, a peer can aim to optimize its lookup performance by minimizing the stretch (or, alternatively: the latency) to the other peers in the network. Achieving only good stretches is of course trivial: a peer simply connects to a large number of other peers in the system. However, since storing and maintaining such a large neighbor set is costly, it is natural that strategic peers also seek to avoid to connect to too many neighbors. In Chapter 9, we investigate this fundamental trade-off between the need to have small latencies and the desire to reduce maintenance overhead that governs the decisions of selfish peers. By a game-theoretic analysis, we derive the *Price of Anarchy* which captures to which extent selfishness degrades the network performance. In other words, we answer the question: how much better would the social welfare be if the selfish players collaborated instead of striving for maximizing their own benefit?

The introduction of micro economic models and game theory in computer science has led to many insights into the reality of not only today's p2p systems but also distributed systems in general (e.g., the Internet which typically connects many different utility-optimizing stake-holders or agents). Over the last years, many aspects of distributed systems have been studied from a game-theoretic point of view. However, the non-cooperation challenge in distributed systems is not restricted to selfishness. These p2p networks are faced with the problem of *malicious* adversaries who try—independently of their own cost—to degrade the utility of the entire system, to attack correctness of certain computations, or to cause endless changes which render the system instable. Chapter 10 introduces a mathematical framework which seeks to combine the two fruitful threads of research game theory and fault-tolerance. We consider a system of selfish individuals whose only goal is to optimize their own benefit, and add malicious players who attack the system in order to deteriorate its overall performance. We ask: what is the impact of the malicious players on a selfish system's efficiency? We exemplify our theory by giving an analysis of a virus inoculation game. In Chapter 11, we make an excursion and show how the framework of Chapter 10 can be applied in the context of *social networks* where players care about the welfare of their contacts. We analyze inoculation strategies in the virus game and compute the so-called *Windfall of Friendship*. We can prove that while the Windfall of Friendship can never be negative compared to purely selfish environments, but it turns out that the social welfare does not increase monotonically with stronger social ties.

To round off and complement our theoretic considerations of Chapter 10 on the effect of malicious peers in p2p networks, Chapter 12 performs a case study investigating the feasibility of malicious attacks in a popular p2p system: the *Kad network*. Since the demise of the Overnet network, Kad has become the only widely used peer-to-peer system based on a distributed hash table, and due to its decentralized nature, it is likely that the Kad's user base will continue growing in numbers over the next few years as. We find that several vulnerabilities exist in the current Kad network. The presented attacks could be used either to hamper the correct functioning of the network itself, to censor contents, or to harm other entities in the Internet not participating in the Kad network such as ordinary web servers. While there are simple heuristics to reduce the impact of some of the attacks, we believe that some of our attacks are inherently difficult to avert and cannot be thwarted easily without some kind of centralized certification and verification authority.

Chapter 8

Free Riding Case Study: BitTorrent

"Now there's BitThief, which is straight-up nefarious and wrong – the client downloads torrents without uploading."
On http://blog.wired.com/ (January 2007)

"Anyhow, bitthief is a client which I've been waiting for so long, I mean.. bitcomet bent the rules but never really broke any of them.. that much Bitthief is an interesting client [...] I wonder how fast this will get banned at every tracker alive. As others have said, this makes bittyrant look like a sunday school boy.."
Another blogger on BitThief (January 2007)

Bram Cohen's BitTorrent protocol [66]—which in 2004, according to *CacheLogic*, represented 35% of the Internet traffic [184]—heralded a paradigm shift as it demonstrated that cooperation can be fostered among peers interested in the same file, and that concentrating on one file is often enough in practice. The fair sharing mechanism of BitTorrent is widely believed to discourage freeloading behavior. Contrary to such belief, this chapter shows that BitTorrent in fact does not provide sufficient incentives to rule out free riding. We have developed our own BitTorrent client *BitThief* that never serves any content to other peers. With the aid of this client, we demonstrate that a peer can download content fast *without uploading any data*. Surprisingly, BitThief always achieves a high download rate, and in some experiments it has even outperformed the *official client*. Moreover, while *seeders* ("altruistic peers") clearly offer the opportunity to freeload, we are even able to retrieve files quickly if we ignore seeders and download solely from other peers that do not possess all pieces of the desired content (*leechers*). This implies that the basic piece exchange mechanism does not effectively restrain peers from freeloading.

Sharing communities are also investigated in this chapter. By banning users with constantly low sharing ratios or by denying them access to the newest torrents available, such communities encourage users to upload more than they download, i.e., to keep their sharing ratio above 1. We will show that sharing communities are particularly appealing for free riders, and that cheating is easy.

We believe that the possibility to freeload which does not come at the cost of a considerably reduced quality of service (e.g., download rate) is attractive for users: not only because wasting more expensive upload bandwidth is avoided, but also because—in contrast to downloading—merely the *distribution* of copyrighted media content such as music or video shared in p2p networks is unlawful in certain countries.

However, as more and more users decide to free ride, the usefulness of a p2p system will decline. Thus, spreading such freeloading clients might prove to be an effective attack for corporations fighting the uncontrolled distribution of their copyrighted material.

8.1 BitTorrent Background

The main mechanisms applied by *BitTorrent* are described in [66]; for additional resources including a detailed technical protocol, the reader is referred to *www.bittorrent.org*. Basically, BitTorrent is a p2p application for sharing files or collections of files. In order to participate in a *torrent download*, a peer has to obtain a *torrent metafile* which contains information about the content of the torrent, e.g. file names, size, tracker addresses, etc. A tracker is a centralized entity that keeps track of all the peers (TCP endpoints) that are downloading in a specific torrent swarm. Peers obtain contact information of other participating peers by announcing themselves to the tracker on a regular basis. The data to be shared is divided into pieces whose size is specified in the metafile (usually a couple of thousand pieces per torrent). A hash of each piece is also stored in the metafile, so that the downloaded data can be verified piece by piece. Peers participating in a torrent download are subdivided into *seeders* which have already downloaded the whole file and which (altruistically) provide other peers with any piece they request, and *leechers* which are still in progress of downloading the torrent. While seeders upload to all peers (in a round robin fashion), leechers upload only to those peers from which they also get some pieces in return. The peer selection for uploading is done by unchoking a fixed number of peers every ten seconds and thus enabling them to send requests. If a peer does not contribute for a while it is choked again and another peer is unchoked instead.

The purpose of this mechanism is to enforce contributions of all peers. However, each leecher periodically *unchokes* a neighboring leecher, transferring some data to this neighboring peer for free (called *optimistic unchoking* in BitTorrent lingo). This is done in order to allow newly joined peers without any pieces of the torrent to *bootstrap*. This unchoking mechanism is one weakness that can be exploited by BitThief.

8.2 BitThief and Analysis

In the following, we will provide evidence that, with some simple tricks, uploading can be avoided in BitTorrent while maintaining a high download rate. In particular, our own client BitThief is described and evaluated. BitThief is written in Java and is based on the official implementation[1] (written in Python, also referred to as *official client* or *mainline client*), and the *Azureus*[2] implementation. We kept the implementation as simple as possible and added a lot of instrumentational code to analyze our client's performance. BitThief does not perform any chokes or unchokes of remote peers, and it never announces any pieces. In other words, a remote peer always assumes that it interacts with a newly arrived peer that has just started downloading. Compared to the official client, BitThief is more aggressive during the startup period, as it re-announces itself to the tracker in order to get many remote peer addresses as quickly as possible. The tracker typically responds with 50 peer addresses per announcement. This parameter can be increased to at most 200 in the announce request, but most trackers will trim the list to a limit of 50. Tracker announcements are repeated at an interval received in the first announce response, usually in the order of once every 1800 seconds. Our client ignores this number and queries the tracker more frequently, starting with a configurable interval and then exponentially backing off to once every half an hour. Interestingly, during all our tests, our client was not banned by any of the trackers and could thus gather a lot of peers. The effect of our aggressive behavior is depicted in Figure 8.1. Finally, note that it would also be possible to make use of the *distributed tracker protocol.*[3] This protocol is useful if the main tracker is not operational. However, thus far, we have not incorporated this functionality into our client.

Having a large number of open connections improves the download rate twofold: first, connecting to more seeders allows our client to benefit more often from their round robin unchoking periods. Second, there will be more leechers in our neighborhood that include BitThief in their periodical optimistic unchoke slot. Opening more connections increases download speed linearly, as remote peers act independently of the number of our open connections. However, note that opening two connections to the same peer does not help, as the official client, Azureus, and presumably all other clients as well immediately close a second connection originating from the same IP address.

Our experiments with BitThief demonstrate that the common belief that the performance will degrade if a large number of TCP connections is maintained simultaneously is unfounded. On the contrary, more connections always help to increase the download rate when using BitThief. The reason why the total number of TCP connections is kept small in BitTorrent might be that a moderate number of connections suffice to saturate the average user's bandwidth when following the real protocol, and no further gain could be achieved by connecting to more peers.

[1]See http://bittorrent.com/.

[2]See http://azureus.sourceforge.net/.

[3]See http://www.bittorrent.org/Draft_DHT_protocol.html.

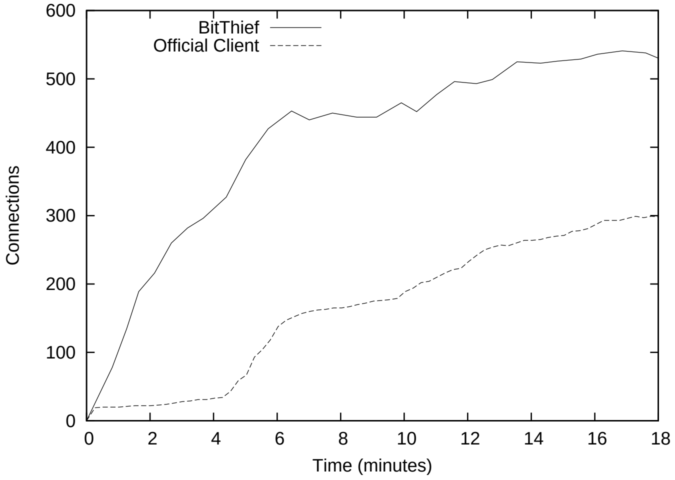


Figure 8.1: Number of open connections over time. In comparison to the official client, BitThief opens connections much faster.

In contrast to other BitTorrent clients, BitThief does not apply the so-called *rarest-first policy*, but uses a simpler piece selection algorithm instead: we fetch whatever we can get. If our client is unchoked by a remote peer, it picks a random missing piece. Our algorithm ensures that we *never* leave an unchoke period unused. Furthermore, just like other BitTorrent clients, we strive to complete the pieces we downloaded partially as soon as possible in order to check them against the hash from the metafile and write them to the harddisk.

8.2.1 Seeders

We first tested the client on several torrents obtained from *Mininova*[4] and compared it to the official client.[5] By default, the official client does not allow more than 80 connections. In order to ensure a fair comparison, we removed this limitation and permitted the client to open up to 500 connections. In a first experiment, we did not impose any restrictions on our client, in particular, BitThief was also permitted to download from seeders. The tests were run on a PC with a public IP address and an open TCP port, so that remote peers could connect to our client. We further blocked all network traffic to or from our university network, as this could bias the measurements. The properties of the different torrents used in this experiment are depicted in

[4]See http://www.mininova.org/.

[5]Official client vers. 4.20.2 (linux source). Obtained from bittorrent.com, used with parameters: `--min_peers 500 --max_initiate 500 --max_allow_in 500`.

	Size	Seeders	Leechers	μ	σ
A	170MB	10518 (303)	7301 (98)	13	4
B	175MB	923 (96)	257 (65)	14	8
C	175MB	709 (234)	283 (42)	19	8
D	349MB	465 (156)	189 (137)	25	6
E	551MB	880 (121)	884 (353)	47	17
F	31MB	N/A (29)	N/A (152)	52	13
G	798MB	195 (145)	432 (311)	88	5

Table 8.1: Characteristics of our test torrents. The numbers in parentheses represent the maximum number of connections BitThief maintained concurrently to the respective peer class and is usually significantly lower than the peer count the tracker provided. μ and σ are the average and standard deviation of the official client's download times in minutes. The tracker of Torrent F did not provide any peer count information. Based on the number of different IP addresses our client exchanged data with, we estimate the total number of peers in this torrent to be more than 340.

Table 8.1. Note that the tracker information is not very accurate in general and its peer count should only be considered a hint on the actual number of peers in the torrent.

The results are summarized in Figure 8.2. As a first observation, note that in every experiment, BitThief succeeded eventually to download the entire file. More interestingly, the time required to do so is often not much longer than with uploading! Exceptions are Torrents E and G, where there are relatively few seeders but plenty of leechers. In that case, it takes roughly four times longer with our client. However, the download came at a large cost for the official client as it had to upload over 3.5GB of data. Torrents A, B and F also offer valuable insights: in those torrents, BitThief was, on average, slightly faster than the official client, which uploaded 232MB in a run of torrent A and 129MB in a run of Torrent B. We conclude that in torrents with many peers, particularly seeders, and in torrents for small files, BitThief seems to have an advantage over the official client, probably due to the aggressive connection opening.

8.2.2 Leechers

In this section, we further constrain BitThief to only download from other leechers. Interestingly, as we will see, even in such a scenario, free riding is possible.

Seeders are identified by the *bitmask* the client gets when the connection to the remote peer is established, and the *have-message* received every time the remote peer has successfully acquired a new piece. As soon as the remote peer has accumulated all pieces, we immediately close the connection. We conducted the tests at the same time as in Chapter 8.2.1 and also used the same torrents. The running times are depicted in Figure 8.3. It does not come

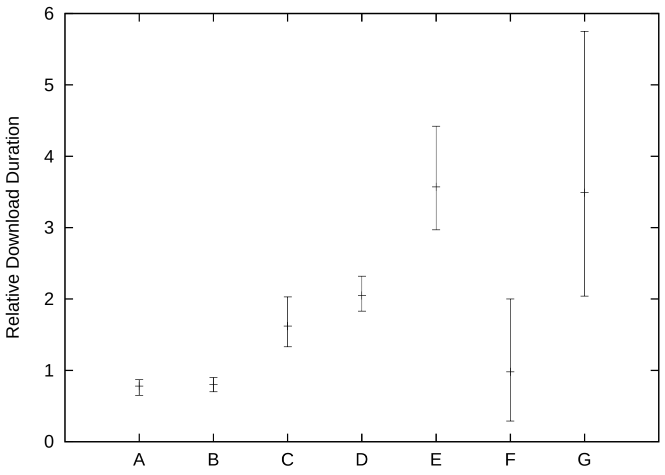


Figure 8.2: Relative download times for six torrents. The download time of the official client is normalized to 1.0. Every torrent was downloaded three times with both clients. The plot shows relative download times with the fastest run at the lower end of the bar, the average running time at the level of the horizontal tick mark, and the slowest run at the upper end of the bar.

as a surprise that the average download time has increased. Nevertheless, we can again see that all downloads finished eventually. Moreover, note that the test is slightly unfair for BitThief, as the official client was allowed to download not only from the leechers, but also from all seeders! In fact, in some swarms only a relatively small fraction of all peers are leechers. For example in Torrent C, merely 15% are leechers, and BitThief can thus download from less than a sixth of all available peers; nevertheless, BitThief only requires roughly 5 times longer than the official client.

We conclude that even without downloading from seeders, BitThief can download the whole torrent from leechers exclusively. Therefore, it is not only the seeders which provide opportunities to free ride, but the leechers can be exploited as well.

8.2.3 Further Experiments

The measurements presented so far have all been obtained through experiments on the Internet and hence were subject to various external effects. For example, in case BitThief was allowed to download from seeders, it sometimes downloaded at a high rate, but then—a few minutes later—the download rate declined abruptly due to a powerful seeder having left the network. In order to get reproducible results, we set up a pet network environment on a

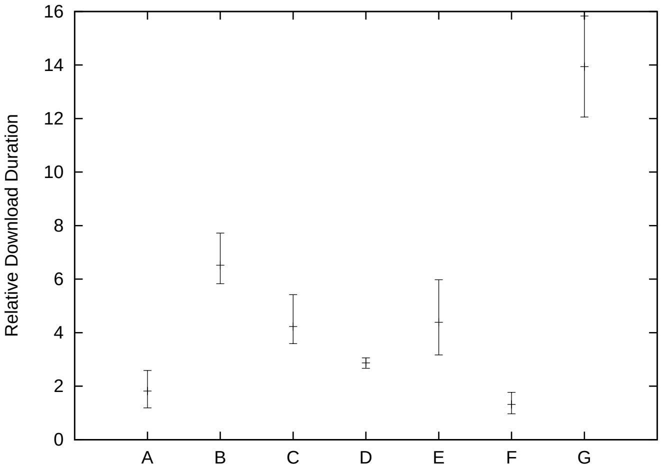


Figure 8.3: Relative download times of BitThief for six torrents *without downloading from any seeders*. The download time of the official client is normalized to 1.0. As in the first experiment, the torrents were downloaded three times with the official client and three times using BitThief restricted to download from leechers only. The bars again represent the same minimum, average, and maximum running times.

host, consisting of a private tracker, a configurable number of official clients as seeders and leechers, and one instance of our own client. We evaluated different scenarios. In the following, our main findings will be summarized briefly.

In scenarios with many seeders and only very few leechers, our client downloads most data from seeders. As the leechers often do not fill up all their upload slots with other leechers, our client is unchoked all the time, yielding a constant download rate.

More interesting are scenarios with a small number of seeders. A fast seeder is able to push data into the swarm at a high rate and all the leechers can reciprocate by sharing the data quickly with their upstreams fully saturated. In this situation, it is difficult for our client to achieve a good downstream: we only get a small share of the seeders' upstream and all the other leechers are busy exchanging pieces between them. Hence, we only profit from the optimistic unchoke slots, which results in a poor performance. However, note that many leechers will turn into seeders relatively soon and therefore our download rate will increase steadily.

A slow seeder is not able to push data fast enough into the swarm, and the leechers reciprocate the newly arrived pieces much faster without filling all their upload slots. Although BitThief cannot profit from the seeders, it

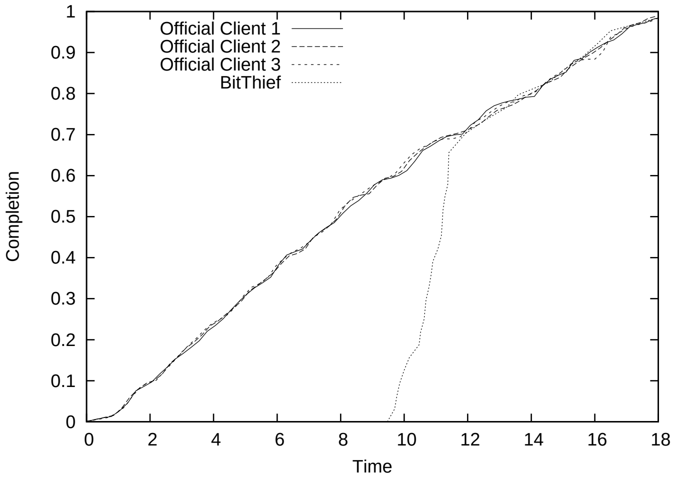


Figure 8.4: Download times for three official clients and one BitThief client in the presence of a slow seeder. BitThief starts downloading 9 minutes later than the other clients, but catches up quickly. Ultimately, all clients finish the download roughly at the same time.

can make use of the leechers' free upload slots. The attainable download rate is similar to the one where there are many seeders. The download rate will go down only when BitThief has collected all pieces available in the swarm. When a new piece arrives, the leechers will quickly exchange it, enabling BitThief to download it as well with almost no delay. An experiment illustrating this behavior is given in Figure 8.4. Note that the execution shown in the figure is quite idealistic, as there are no other leechers joining the torrent over time.

In summary, the results obtained from experiments on the Internet have been confirmed in the experiments conducted in our pet network.

8.2.4 Exploiting Sharing Communities

Finding the right torrent metafile is not always an easy task. There exist many sites listing thousands of torrents (e.g., Mininova), but often the torrents' files are not the ones mentioned in the title or are of poor quality. Therefore, a lot of *sharing communities* have emerged around BitTorrent. These communities usually require registration on an invitation basis or with a limit on the number of active users. Finding good quality torrents in these communities is much more convenient than on public torrent repositories. Sharing communities usually encourage their users to upload at least as much data as they download, i.e., to keep their sharing ratio above 1. This is achieved by banning users with constantly low sharing ratios or by denying them access to the newest torrents available.

Andrade et al. [16] studied these communities and analyzed how sharing ratio enforcement influences seeding behavior. The authors find that seeders are staying in a torrent for longer periods of time, i.e., typically the majority of peers are seeders. These communities thus exhibit ideal conditions for BitThief, provided that we can find ways to access and stay in this communities without uploading.

We have found that this can often be done by simply pretending to upload. The community sites make use of the tracker announcements which every client performs regularly. In these announcements the client reports the current amount of data downloaded and uploaded. These numbers are stored in a database and used later on to calculate the sharing ratios. The tracker typically does not verify these numbers, although, in our opinion, it would be possible to expose mischievous peers: for instance, in a torrent with 100 seeders and just one leecher, it looks suspicious if the leecher is constantly announcing large amounts of uploaded data. Alternatively, the sum of all reported download and upload amounts could be analyzed over different torrents and time periods, in order to detect and ban dishonest peers.

The tracker can also be cheated easily: clients can announce bogus information and fake peers so that the tracker's peer list fills up with dozens of clients which do not exist. The seeder and leecher counts reported by the tracker can therefore be misleading as there are usually not that many real peers downloading a given torrent. Even worse, peers asking a tracker for other peers can get a lot of invalid or stale information, which makes torrent starts slow.[6] The technique of faking tracker announcements has been used in a couple of torrents in our tests and we now have a sharing ratio of 1.4 on *TorrentLeech*[7] without ever uploading a single bit.

An example which emphasizes how dramatic the difference between a community internal and an external download can be, is given in Figure 8.5. We used a torrent that was published on TorrentLeech approximately 12 hours before conducting this experiment and looked for the same one on Mininova, where it had appeared 4 hours earlier. The torrent was 359MB in size on TorrentLeech and slightly smaller (350MB) on Mininova. We first downloaded the torrent three times from Mininova, then three times from TorrentLeech. The Mininova runs took 32/32/37 minutes, while on TorrentLeech the runs completed in 7:25/7:08/7:08 minutes, respectively. This is more than four times faster. Considering that there were only 25 (24 seeders, one leecher) peers in the TorrentLeech swarm and more than 834 (531 seeders, 303 leechers) peers in the other swarm, this is surprising.

As far as the individual contributions of the peers are concerned, we observed the following. While BitThief tends to benefit more from certain peers, generally seeders, in public torrents, a much larger fraction of all peers

[6] An alternative is used by recent BitTorrent clients: a distributed tracker protocol which manages the torrent swarm.

[7] See http://torrentleech.org/.

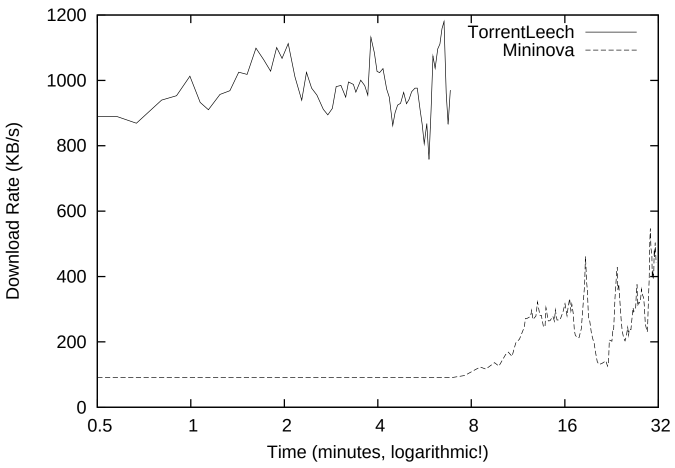


Figure 8.5: BitThief's download speed: comparison between a community version of a torrent and one found on Mininova.

provides a considerable share of the file in sharing communities, and the distribution across peers is more balanced. This is probably due to the community peers' desire to boost their sharing ratios by uploading as much as possible. An experiment illustrating this point is depicted in Figure 8.6.

8.3 Sophisticated Exploits

While simple tricks often yield a good performance, BitTorrent has proved to be quite robust against certain more sophisticated attacks.

First, we have investigated an exploit proposed in [150] which truly violates the BitTorrent protocol: the selfish client announces pieces as being available even if it does not possess them. If such an unavailable piece is requested by a remote peer, the client simply sends random data (garbage). As only the integrity of whole pieces can be checked, the remote peer cannot verify the subpiece's correctness. Note that this behavior cannot be considered free riding in the pure sense, but it is a strategy that does not require to upload any *valid* user data.

In a first implementation, all requests are answered by uploading entire garbage pieces. As has already been pointed out in [150], this approach is harmful: both the official client and Azureus store information from whom they have received subpieces and will thus immediately ban our IP address once the hash verification fails.[8] Consequently, we have tried to answer all

[8]Note that an appealing solution would be to fake entire pieces by using contents

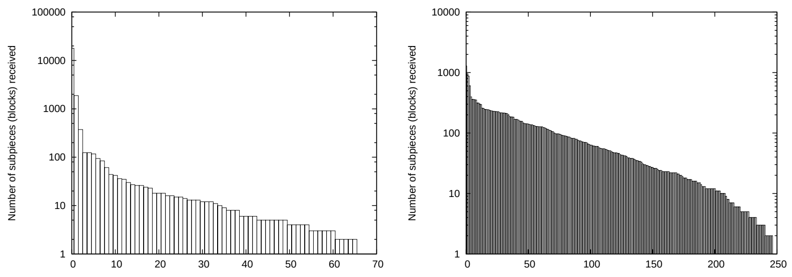


Figure 8.6: Number of blocks obtained by BitThief from different peers. The file size was 350 MB. On Mininova (*left*) and on TorrentLeech (*right*), BitThief connected to 309 and 349 peers, respectively. In the community network, the distribution is more balanced and BitThief is able to download from much more peers.

requests for a piece except for one subpiece, which would force the remote peer to get that subpiece from a different peer. The idea is that the remote peer cannot tell which peer uploaded the fake data, as it might as well be the other peer which only supplied one subpiece. While the official client can indeed be fooled this way, Azureus is smarter and uses an interesting approach: once it has determined that the piece is not valid, it looks up from which peer it received most subpieces. The piece is then reserved for that peer, and Azureus aims at fetching all remaining subpieces from the same peer. When refusing to answer these requests, the connection stalls, and eventually our IP address is banned. We have tried several tricks to circumvent these problems, but came to the conclusion that uploading random garbage, in any way, does not improve performance.

When establishing connections, peers inform each other about their download status by sending a list of pieces that they have already successfully downloaded. While the connection is active, peers send messages to each other for each new piece they downloaded. Therefore, a peer always knows the progress of its neighbors. We sought to measure the influence that this information has on a remote peer. Currently, BitThief sends an empty list of available pieces during connection setup and it does not inform the remote peer about any new pieces it acquires. We tried announcing different percentages of all the pieces at the beginning of the connection, but our experiments showed that the performance is independent of the percentage, as long as not all of the pieces are available. However, announcing 100% of the pieces has disastrous consequences, as the remote peer considers BitThief a seeder and therefore does not respond to any piece requests.

BitThief profits from the optimistic unchoke slots of leechers and from

yielding the same hash values. Unfortunately, however, the computation of such SHA-1 hash collisions is expensive and would yield huge tables which cannot be stored in today's databases.

the round robin unchoke scheme of seeders. Thus, a client could possibly increase the chance of being unchoked by being present in the remote peer's neighborhood more than once. This is known as a *Sybil attack* [81]. However, this attack involves opening two or more connections to a remote peer. Both the official client and Azureus prevent such behavior. If multiple IP addresses are available, it would be an easy task to extend the client in a way to fake two entities and trick remote peers. The peers would gladly open a connection to both external addresses and thus our download rate might increase up to twofold.

8.4 Concluding Remarks

This chapter has demonstrated that the prevalent BitTorrent system is not robust to cooperation attacks today as it can cheated by strategic peers in several ways. We believe that the presence of a large fraction of free riders can be harmful to such a system's performance. Of course, only time can tell how many BitTorrent users will indeed act selfishly in the future. BitThief has received quite some attention: the paper itself has been downloaded from our research group's web server more than 100 times per day on average in 2007, the effects have been discussed in many blogs, and first Wikipedia articles and YouTube tutorials have appeared on the web. While many bloggers worried about the consequences BitThief might have on the file sharing performance, some users announced to appreciate that the client does not upload any data. Interestingly, BitThief has also been recommended in the Mininova FAQ[9]:

> *" **Is downloading torrent files from Mininova illegal?** [...] In some other countries, it is allowed to download copyrighted material, as long as you don't upload. If this is the case, we advise you to use the BitThief client. [...] "*

Up to now, BitThief's user base is still small. Figure 8.7 plots the number of accessed torrents and the total number of different IP addresses using BitThief per month from January 2007 to December 2007. However, we believe that this does not necessarily imply that people are eager to contribute in p2p networks. We have implemented BitThief mainly to give a proof-of-concept and—apart from publishing our measurement results—did not promote the software. The client still lacks many desirable features and has a simple GUI. Moreover, it is well-known that BitThief occasionally transmits statistical data to our server. Although this data merely contains anonymous and hashed information about the time required to download files of any size, it might be a good reason for some people not to install BitThief.

One possible thread of future research is the improvement of our client to incorporate other selfish attacks, e.g., *collusion*, or to allow a limited amount of upload data. Moreover, current trends such as *ISP caching*[10] could also

[9]See http://www.mininova.org/faq.

[10]See CacheLogic Press Release http://www.cachelogic.com/home/pages/news/pr070806.php.

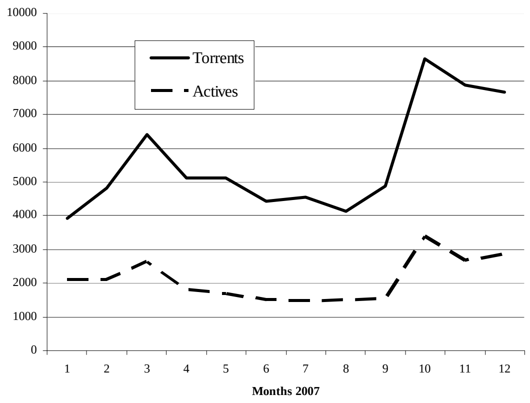


Figure 8.7: Number of accessed torrents (*Torrents*) and number of different IP addresses (*Actives*) using BitThief per month during the year 2007. (BitThief was released in January 2007.)

introduce new potential exploits.

More interesting, however, would be to extend BitThief such that cooperation among peers is truly enforced. For this purpose, the *Fast Extension*[11] might serve as a promising starting point. The challenge is to find a mechanism that applies some kind of tit-for-tat algorithm for older peers in the system, while at the same time efficiently solving the *bootstrap problem* [189] of newly joining peers: as these new peers inherently do not have any data to share, they must be provided with some "venture capital". Some of our first findings regarding a more efficient tit-for-tat algorithm can be found in [156]: our approach makes use of source coding techniques. Finally, our fairness scheme for live-streaming in Pulsar is also inspired by the tit-for-tat scheme and can be found in [152].

[11]See http://bittorrent.org/fast_extensions.html.

Chapter 9

Impact of Selfish Players

This chapter investigates the impact of selfish behavior in unstructured peer-to-peer topologies. Concretely, we study the quality of the network topologies which result if peers selfishly select to which other peers they connect. *Game theory* provides tools to analyze such strategic behavior.

One contribution will be the computation of the *Price of Anarchy* of p2p overlay creation, which is the ratio between an optimal solution—where participants are controlled by a benevolent master puppeteer (or by one of the philosopher kings in Plato's utopian *Kallipolis*)—compared to a solution generated by peers that act in an egoistic manner, optimizing their individual benefit.

The importance of studying the Price of Anarchy in peer-to-peer systems stems from the fact that it quantifies the possible degradation caused by selfishness. Specifically, a low Price of Anarchy indicates that a system does not require an incentive-mechanism (such as *tit-for-tat*), because selfishness does not overly bog down the overall system performance. If the Price of Anarchy is high, however, specific cooperation incentives (whose goals are to reduce the Price of Anarchy) need to be enforced in order to ensure that the system can perform efficiently. Hence, in peer-to-peer systems the Price of Anarchy is a measure that helps explaining the necessity (or non-necessity) of cooperation mechanisms.

We will first show that the topologies of selfish, unstructured p2p systems can be much worse than in a scenario in which peers collaborate. More precisely, we show that the Price of Anarchy is $\Theta(\min(\alpha, n))$, where α is a parameter that captures the tradeoff between lookup performance (low stretches) and the cost of neighbor maintenance, and n is the number of peers in the system. Thereby, the upper bound $O(\min(\alpha, n))$ holds for peers located in *arbitrary* metric spaces, including the popular *growth-bounded* and *doubling metrics*. On the other hand, intriguingly, this bound is tight even in such a simple metric space as the 1-dimensional *Euclidean space*. As a second contribution, we prove that the topology of a static peer-to-peer system consisting of selfish peers may never converge to a stable state. That is, links may continuously change even in environments without churn, causing the network to be inherently instable. Finally, we consider the complexity of Nash equilibria. We show that deciding whether there exists a pure Nash equilibrium in a given network is **NP**-hard. Consequently, it is infeasible in practice to determine if a p2p network of selfish peers can stabilize.

9.1 A P2P Network Creation Game

We model the peers of a p2p network as points in a *metric space* $\mathcal{M} = (V, d)$, where $d : V \times V \rightarrow [0, \infty)$ is the *distance function* which describes the underlying latencies between all pairs of peers. The effects of selfish peer behavior is studied from a game-theoretic perspective. We consider a set of n peers $V = \{\pi_0, \pi_1, \ldots, \pi_{n-1}\}$. A peer can choose to which subset of other peers it wants to store pointers (IP addresses). Formally, the *strategy space* of a peer π_i is given by $S_i = 2^{V\setminus\{\pi_i\}}$, and we will refer to the actually chosen links as π_i's *strategy* $s_i \in S_i$. We say that π_i *maintains or establishes a link* to π_j if $\pi_j \in s_i$. The combination of all peers' strategies, i.e., $s = (s_0, ..., s_{n-1}) \in S_0 \times \cdots \times S_{n-1}$, yields a (directed) graph $G[s] = (V, \cup_{i=0}^{n-1}(\{\pi_i\} \times s_i))$, which describes the resulting p2p topology.

Selfish peers exploit *locality* in order to maximize their lookup performance. Concretely, a peer aims at minimizing the *stretch* to all other peers. The stretch between two peers π and π' is defined as the shortest distance between π and π' using the links of the resulting p2p topology G divided by the direct distance, i.e., for a topology G, $stretch_G(\pi, \pi') = d_G(\pi, \pi')/d(\pi, \pi')$. Clearly, it is desirable for a peer to have low stretch to other peers in order to keep its latency small. By establishing a link to all peers in the system, a peer reaches every peer with minimal stretch 1, and the potential lookup performance is optimal. However, storing and especially maintaining a large number of links is expensive. Therefore, the individual cost $c_i(s)$ incurred at a peer π is composed not only of the stretches to all other peers, but also of its *degree*, i.e., the number of its neighbors:

$$c_i(s) = \alpha \cdot |s_i| + \sum_{i \neq j} stretch_{G[s]}(\pi_i, \pi_j).$$

Note that this cost function captures the classic p2p trade-off between the need to minimize latencies and the desire to store and maintain only few links, as it has been addressed by many existing systems, for example Pastry [207]. Thereby, the relative importance of degree costs versus stretch costs is expressed by the parameter α.

The objective of a selfish peer is to minimize its individual cost. In order to evaluate the topologies constructed by selfish peers—and compare them to the topologies achieved by collaborating peers—we use the notion of a *Nash equilibrium.* A p2p topology constitutes a Nash equilibrium if no peer can reduce its individual cost by changing its set of neighbors given that the connections of all other peers remain the same. More formally, a (pure) Nash equilibrium is a combination of strategies s such that, for each peer π_i, and for all alternative strategies s' which differ only in the i^{th} component (different neighbor sets for peer π_i), $c_i(s) \leq c_i(s')$. This means that in a Nash equilibrium, no peer has an incentive to change its current set of neighbors, that is, Nash equilibria are *stable*.

While peers try to minimize their individual cost, the system designer is interested in a good overall quality of the p2p network. The *social cost* is the sum of all peers' individual costs, i.e.,

$$C(G) = \sum_i c_i = \alpha|E| + \sum_{i \neq j} stretch_G(\pi_i, \pi_j).$$

The lower this social cost, the better is the system's performance.

Determining the parameter α in real unstructured peer-to-peer networks is an interesting field for study. As mentioned, α measures the relative importance of low stretches compared to the peers' degrees, and thus depends on the system or application: for example, in systems with many lookups where good response times are vital, α is smaller than in distributed archival storage systems consisting mainly of large files. In the sequel, we will denote the link and stretch costs by

$$C_E(G) = \alpha|E| \quad \text{and} \quad C_S(G) = \sum_{i \neq j} stretch_G(\pi_i, \pi_j).$$

Typically, a given distribution of peers in a metric space can result in different Nash equilibria, depending on the order in which peers change their links. To gain an understanding of the impact of selfishness on the social cost, we are particularly interested in the social cost of the *worst* possible Nash equilibrium. That is, we study topologies in which no selfish peer has an incentive to change its neighbors, but in which all peers together could be much better off if they collaborated. More precisely and using the terminology of game theory, we are interested in the *Price of Anarchy*, the ratio between the social cost of the worst Nash equilibrium and the social cost of the optimal topology.

9.2 Price of Anarchy

The Price of Anarchy is a measure to bound the degradation of a globally optimal solution caused by selfish individuals. In this section, we show that the topologies created by selfish peers deteriorate more (compared to collaborative networks) as the cost of maintaining links becomes more important (larger α). Concretely, in Chapter 9.2.1 we prove that for *arbitrary* metric spaces—thus, including the important and well-studied *growth-bounded* [123] and *doubling* (e.g. [59]) metrics—the Price of Anarchy never exceeds $O(\min(\alpha, n))$. We then show in Chapter 9.2.2 that this bound is tight even in the "simplest" metric space, the 1-dimensional Euclidean space, where there exist Nash equilibria with a Price of Anarchy of $\Omega(\min(\alpha, n))$.

9.2.1 Upper Bound

Assume the most general setting where n peers are arbitrarily located in a given metric space $\mathcal{M}$, and consider a peer π which has to find a suitable neighbor set. Clearly, the *maximal* stretch from π to any other peer π' in the system is at most $\alpha + 1$: if $stretch(\pi, \pi') > \alpha + 1$, π could establish a direct link to π', reducing the stretch from more than $\alpha + 1$ to 1, while incurring a link cost of α. Therefore, in any Nash equilibrium, no stretch exceeds $\alpha + 1$. Because there are at most $n(n-1)$ directed links (from each peer to all remaining peers), the social cost of a Nash equilibrium is $O(\alpha n^2)$. In the social optimum on the other hand, all stretches are at least 1 and there must be at least $n - 1$ links in order to keep the topology connected. This lower bounds the optimal social cost by $\Omega(\alpha n + n^2)$ and yields the following result.

Theorem 9.1. *For any metric space $\mathcal{M}$, the Price of Anarchy is $O(\min(\alpha, n))$.*

Theorem 9.1 implies that if the relative importance of the peers' stretch is large, the Price of Anarchy is small. That is, for small α, the selfish peers have an incentive to establish links to many other peers, while also the optimal network is highly connected.

9.2.2 Lower Bound

We now show that there are p2p networks in which the Price of Anarchy is as bad as $\Omega(\min(\alpha, n))$, which implies that the upper bound of Chapter 9.2.1 is asymptotically tight. Intriguingly, the Price of Anarchy can deteriorate to $\Theta(\min(\alpha, n))$ even if the underlying latency metric describes a simple 1-dimensional Euclidean space.

Consider the topology G in which peers are located on a line, and the distance (latency) between two consecutive peers increases exponentially towards the right. Concretely, peer i, for i from 1 to n, is located at position $\alpha^{i-1}/2$ if i is odd, and at position α^{i-1} if i is even. The peers of G maintain links as follows: all peers have a link to their nearest neighbor on the left. Odd peers additionally have a link to the second nearest peer on their right. After proving that G constitutes a Nash equilibrium, we derive the lower bound on the Price of Anarchy by computing the social cost of this topology.

Lemma 9.2. *The topology G forms a Nash equilibrium for $\alpha \geq 3.4$.*

Proof. We distinguish between even and odd peers. For both cases, we show that no peer has an incentive to deviate from its strategy.

Case even peers: Every even peer i needs to link to at least one peer on its left, otherwise i cannot reach the peers $j < i$. A connection to peer $i - 1$ is optimal, as the stretch to all peers $j < i$ becomes 1. Observe that every alternative link to the left would imply a larger stretch to at least one peer on the left without reducing the stretch to peers on the right. Furthermore, i cannot reduce the distance to any—neither left nor right—peer by adding further links to the left. Hence, it only remains to show that i cannot benefit from adding more links to the right.

By adding a link to the right, peer i shortens the distance to *all* peers on the right. However, we show that the cost reduction per peer decreases as a geometric series, and any such link to the right would strictly increase i's costs. We consider two cases: i linking to an odd peer on the right, and to an even peer on the right.

Link to an odd peer: Consider the benefit of i adding a link to its odd neighbor $i+1$. For an odd peer $j > i$, we define the *benefit* $B_{i,j}$ as the stretch cost reduction caused by the addition of the link $(i, i+1)$. We have, for $i \geq 2$,

$$\begin{aligned} B_{i,j} &= stretch_{old}(i,j) - stretch_{new}(i,j) = \frac{d(i,i-1)+d(i-1,j)}{d(i,j)} - \frac{d(i,j)}{d(i,j)} \\ &= \frac{\alpha^{i-1} - \frac{1}{2}\alpha^{i-2} + \frac{1}{2}\alpha^{j-1} - \frac{1}{2}\alpha^{i-2}}{\frac{1}{2}\alpha^{j-1} - \alpha^{i-1}} - 1 = \frac{2\alpha^{i-1} - \alpha^{i-2}}{\frac{1}{2}\alpha^{j-1} - \alpha^{i-1}} = \frac{2 - \frac{1}{\alpha}}{\frac{1}{2}\alpha^{j-i} - 1}. \end{aligned}$$

Similarly, the savings $B_{i,j}$ for an even peer $j > i$ and $i \geq 2$ amount to $B_{i,j} = stretch_{old}(i,j) - stretch_{new}(i,j) = (d(i,i-1) + d(i-1,j+1) + d(j+1,j))/(d(i,j))$ $-(d(i,j+1) + d(j+1,j))/(d(i,j)) = (\alpha^{i-1} - \alpha^{i-2} + \alpha^{j} - \alpha^{j-1})/(\alpha^{j-1} - \alpha^{i-1}) - (\alpha^{j} - \alpha^{i-1} - \alpha^{j-1})/(\alpha^{j-1} - \alpha^{i-1}) = (2\alpha^{i-1} - \alpha^{i-2})/(\alpha^{j-1} - \alpha^{i-1}) = (2 - \frac{1}{\alpha})/(\alpha^{j-i} - 1)$. Hence, for all $\alpha \geq 3.4$, the total savings B_i for peer i are less than

$$\begin{aligned} B_i &= \sum_{\text{odd } j > i} B_{i,j} + \sum_{\text{even } j > i} B_{i,j} \leq \sum_{\delta=1}^{\infty} \frac{2 - \frac{1}{\alpha}}{\frac{1}{2}\alpha^{2\delta-1} - 1} + \sum_{\delta=1}^{\infty} \frac{2 - \frac{1}{\alpha}}{\alpha^{2\delta} - 1} \\ &\underset{(\alpha \geq 3)}{\leq} \sum_{\delta=1}^{\infty} \frac{2 - \frac{1}{\alpha}}{\frac{1}{2}\alpha^{2\delta-2}} + \sum_{\delta=1}^{\infty} \frac{2 - \frac{1}{\alpha}}{\alpha^{2\delta-1}} = \left(2 - \frac{1}{\alpha}\right) \sum_{\delta=1}^{\infty} \left(\frac{1}{\frac{1}{2}\alpha^{2\delta-2}} + \frac{1}{\alpha^{2\delta-1}} \right) \\ &= \left(2 - \frac{1}{\alpha}\right)\left(\frac{2\alpha^2}{\alpha^2 - 1} + \frac{\alpha}{\alpha^2 - 1} \right) = \frac{4\alpha^2 - 1}{\alpha^2 - 1} \underset{(\alpha \geq 3.4)}{<} \alpha + 1. \end{aligned}$$

Therefore, the construction of link $(i, i+1)$ would be of no avail (benefit smaller than cost). The benefit of alternative or additional links to odd neighbors on the right is even smaller.

Link to an even peer: A link to an even peer $j > i$ entails a stretch 1 to the corresponding peer instead of $stretch_{old}(i,j) = (\alpha^{j} - \alpha^{j-1} + \alpha^{i-1} - \alpha^{i-2})/(\alpha^{j-1} - \alpha^{i-1}) < \alpha + 1$ for $\alpha > 2$. However, the stretch from i to all other peers remains unchanged, since the path $i \rightsquigarrow (i-1) \rightsquigarrow (i+1)$ is shorter than $i \rightsquigarrow (i+2) \rightsquigarrow (i+1)$: $\alpha^{i-1} - \frac{1}{2}\alpha^{i-2} + \frac{1}{2}\alpha^{i} - \frac{1}{2}\alpha^{i-2} < \alpha^{i+1} - \alpha^{i-1} + \alpha^{i+1} - \frac{1}{2}\alpha^{i}$ for $\alpha > 1$. Therefore, an even peer i has no incentive to build links to any even peer on its right.

Case odd peers: An odd peer i needs to link to peer $i-1$, otherwise there is no connection to $i-1$ and the stretch from i to $i-1$ is infinite. Moreover, if the link $(i, i-1)$ is established, $stretch(i,j) = 1$ for all $j < i$. Therefore, peer i does not profit from building additional or alternative links to the left.

It remains to study links to the right. In order to reach all peers with a finite stretch, peer i needs a link to some peer $j \geq i+2$. In the following, we first show that peer i can always benefit from a link $(i, i+2)$, independently of additional links to the right. Secondly, we prove that if i has a link $(i, i+2)$, it has no incentive to add further links.

Assume peer i has no direct link to peer $i+2$. Then, $\mathit{stretch}(i, i+2) \geq (2\alpha^{i+2} - \frac{1}{2}\alpha^{i-1} - \frac{1}{2}\alpha^{i+1})/(\frac{1}{2}\alpha^{i+1} - \frac{1}{2}\alpha^{i-1}) > \alpha + 1$. Hence, no matter which links it already has, peer i can benefit by additionally pointing to peer $i+2$. On the other hand, if i maintains the link $(i, i+2)$, any other links to the right only reduce i's gain. For *odd* peers, this is obvious, since the corresponding stretches are already optimal. A link (i, j) to some *even* peer $j > i$ only improves the stretch to peer j itself, but not to other peers. The stretch to peer j becomes 1 instead of $\mathit{stretch}_{old}(i, j) = (\frac{1}{2}\alpha^{j+1} - \frac{1}{2}\alpha^{i-1} + \frac{1}{2}\alpha^{j+1} - \alpha^j)/(\alpha^j - \frac{1}{2}\alpha^{i-1}) = (\alpha^{j+1} - \alpha^j - \frac{1}{2}\alpha^{i-1})/(\alpha^j - \frac{1}{2}\alpha^{i-1}) < \alpha + 1$ for $\alpha > 0$. Thus, also this link would increase i's costs. □

Lemma 9.3. *The social cost $C(G)$ of the topology G is $C(G) \in \Theta(\alpha n^2)$.*

Proof. The topology G has $n-1$ links pointing to the left and $\lfloor n/2 \rfloor$ links pointing to the right. Hence, the total link costs are

$$C_E(G) = \alpha\left[(n-1) + \lfloor n/2 \rfloor\right] \in \Theta(\alpha n).$$

It remains to compute the costs of the stretches.

The stretch from an odd peer i to an even peer $j > i$ is $\mathit{stretch}(i, j) = (\alpha^j - \alpha^{j-1} - \frac{1}{2}\alpha^{i-1})/(\alpha^{j-1} - \frac{1}{2}\alpha^{i-1}) > (\frac{1}{2}\alpha^j - \frac{1}{2}\alpha^{i-1})/(\alpha^{j-1} - \frac{1}{2}\alpha^{i-1}) > \frac{1}{2}\alpha$ for $\alpha > 2$. Thus, the sum of the stretches of an odd peer i is

$$\begin{aligned} C_S(i) &= \sum_{j<i} \mathit{stretch}(i,j) + \sum_{j>i} \mathit{stretch}(i,j) \\ &> (i-1) + \frac{1}{2}\alpha \left\lfloor \frac{n-i-1}{2} \right\rfloor + \left\lfloor \frac{n-i}{2} \right\rfloor . \end{aligned}$$

The stretch between two even peers i and j is $\mathit{stretch}(i, j) = (\alpha^j - \alpha^{j-1} + \alpha^{i-1} - \alpha^{i-2})/(\alpha^{j-1} - \alpha^{i-1}) > (\frac{1}{2}\alpha^j - \frac{1}{2}\alpha^{i-1})/(\alpha^{j-1} - \alpha^{i-1}) > \frac{1}{2}\alpha$ for $j > i$ and all $\alpha > 2$. Thus, the stretch costs are at least

$$C_S(i) > (i-1) + \frac{1}{2}\alpha \left\lfloor \frac{n-i-1}{2} \right\rfloor - 1 + \left\lfloor \frac{n-i-1}{2} \right\rfloor .$$

Adding up the stretches of odd and even peers yields a lower bound on the total stretch costs:

$$\begin{aligned} C_S(G) &= \sum_{i\ even} C_S(i) + \sum_{i\ odd} C_S(i) \\ &> \frac{n(n-2)}{2} + \alpha\frac{(n-3)(n-2)-n}{8} + \frac{(n-1)(n-2)}{4} \in \Omega(\alpha n^2). \end{aligned}$$

Thus, in combination with Theorem 9.1, it follows that $C_S(G) \in \Theta(\alpha n^2)$. The proof is concluded by combining link and stretch costs, $C(G) = C_E(G) + C_S(G) \in \Theta(\alpha n^2)$. □

Theorem 9.4. *The Price of Anarchy of the peer topology G is $\Theta(\min(\alpha, n))$.*

Proof. The upper bound follows directly from the result obtained in Theorem 9.1. As for the lower bound, if $\alpha < 3.4$, the theorem holds because $\Theta(\min\{\alpha, n\}) \in O(1)$ in this case. By Lemma 9.2, the topology G constitutes a Nash equilibrium for $\alpha \geq 3.4$. Moreover, by Lemma 9.3, the social cost of G are in order of $\Theta\left(\alpha n^2\right)$. In the following, we prove that the optimal social cost is upper bounded by $O(n^2 + \alpha n)$ from which the claim of the theorem follows by dividing the two expressions.

Consider again the peer distribution of G, and assume that there are no links. If every peer connects to the nearest peer to its left and to the nearest peer to its right, there are $2(n-1)$ links, and all stretches are 1. Thus, the social cost of this resulting topology $\widetilde{G}$ is $C(\widetilde{G}) = \alpha \cdot 2(n-1) + n(n-1) \in O(n^2 + \alpha n)$. The optimal social cost is at most the social cost of $\widetilde{G}$. □

9.3 Existence of Nash Equilibria

In this section, we show that a system of selfish peers may never converge to a stable state, even in the absence of churn, mobility, or other sources of dynamics. Interestingly, this result even holds if we assume latencies to form simple metric spaces, such as a 2-dimensional Euclidean space. Specifically, there may not exist a pure Nash equilibrium for certain p2p networks in our "locality game".

Theorem 9.5. *Regardless of the magnitude of α, there are metric spaces $\mathcal{M}$, for which there exists no pure Nash equilibrium, i.e., certain p2p networks cannot converge to a stable state. This is the case even if $\mathcal{M}$ is a 2-dimensional Euclidean space.*

Instead of presenting the formal proof (which will be implicit in the proof of Theorem 9.6), we attempt to highlight the main idea only. Assume that the parameter α is a multiple of 0.6, i.e., $\alpha_k = 0.6k$ for an arbitrary integer $k > 0$. Given a specific k, the 2-dimensional Euclidean instance I_k of Figure 9.1 has no pure Nash equilibrium. Specifically, I_k constitutes a situation in which there are peers $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$ that continue to deviate to a better strategy ad infinitum, i.e., the system cannot converge.

The n peers of instance I_k are grouped into five clusters Π_1, Π_2, Π_a, Π_b, and Π_c, each containing $k = n/5$ peers. Within a cluster, peers are located equidistantly in a line, and each cluster's diameter is ϵ/n, where $\epsilon > 0$ is an arbitrarily small constant. The *inter-cluster distance* $d(\Pi_i, \Pi_j)$ between Π_i and Π_j is the minimal distance between any two peers in the two clusters. Distances not explicitly defined in Figure 9.1 follow implicitly from the constraints imposed by the underlying Euclidean plane.

The proof unfolds in a series of lemmas that characterize the structure of the resulting topology $G[s]$ if the strategies s form a Nash equilibrium in I_k. First, it can be shown that in $G[s]$, two peers in the same cluster are always connected by a path that does not leave the cluster. Secondly, it can be

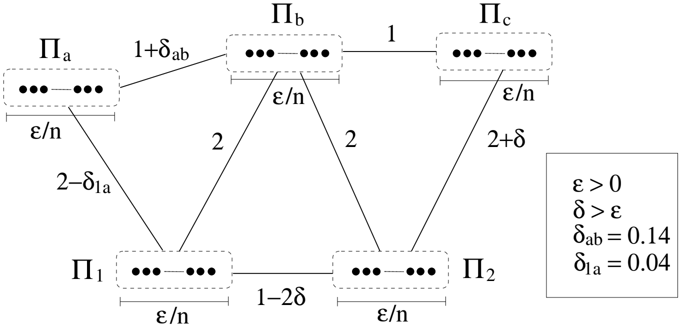


Figure 9.1: Instance I_k has no pure Nash equilibrium when $\alpha = 0.6k$, where $k = n/5$. The number of peers in each cluster is k.

shown that there exists exactly one link in both directions between clusters Π_a and Π_b, Π_b and Π_c, as well as between Π_1 and Π_2. A third structural characteristic of any Nash equilibrium is that for every i and j, there is *at most one* directed link from a cluster Π_i to peers in a cluster Π_j.

To preserve connectivity, some peers in Π_1 and Π_2 must have links to top-peers. Based on the aforementioned observations, the set of possible strategies can further be narrowed down as follows.

i) Neither peers in Π_1 nor Π_2 select three links to top-peers.

ii) There exists a peer $\pi_1 \in \Pi_1$ that establishes a link to Π_a.

iii) There is exactly one link from cluster Π_2 to either cluster Π_b or Π_c, but there is no link to Π_a.

Correctness of all three properties is proven by verifying that there exists some peer $\pi_1 \in \Pi_1$ or $\pi_2 \in \Pi_2$ that has an incentive to change its strategy in case the property is not satisfied. If, for instance, there are two peers $\pi_2, \pi_2' \in \Pi_2$ that simultaneously maintain links to Π_b and Π_c (thus violating Case iii)), π_2' can lower its costs by dropping its link to Π_c. This holds because the sum of the stretches $\sum_{\pi_c \in \Pi_c} stretch(\pi_2', \pi_c)$ entailed by the indirection $\pi_2' \rightsquigarrow \pi_2 \rightsquigarrow \Pi_b \rightsquigarrow \Pi_c$ does not justify the additional cost α.

It can be shown that only the six structures depicted in Figure 9.2 remain valid candidates for Nash topologies. In each scenario, however, at least one peer benefits from deviating from its current strategy.

Case 1: In this case, a peer $\pi_1 \in \Pi_1$ can reduce its cost by adding a link to a peer in Π_b.

Case 2: If the only outgoing link from Π_1 to a top-cluster is to cluster Π_a, the peer $\pi_2 \in \Pi_2$ maintaining the link to Π_c can be shown to profit from

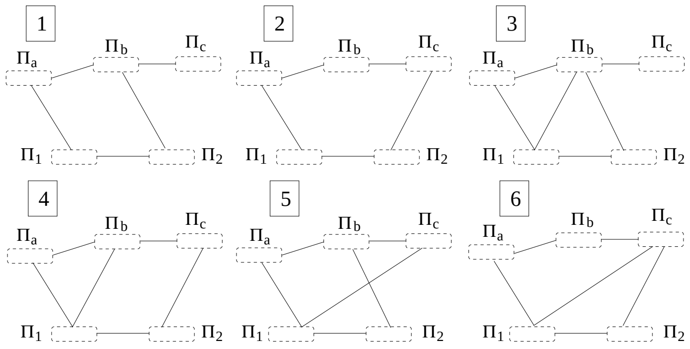


Figure 9.2: Candidates for a Nash equilibrium.

switching its link from Π_c to Π_b.

Case 3: The availability of the link from Π_1 to Π_b changes the optimal choice of the above mentioned peer $\pi_2 \in \Pi_2$. Unlike in the previous case, π_2 now prefers linking to Π_c instead of Π_b.

Case 4: Due to the existence of a link from a peer $\pi_2 \in \Pi_2$ to Π_c, the peer $\pi_1 \in \Pi_1$ with the link to Π_b has an incentive to drop this link and instead use the detours via Π_2 and Π_a to connect to Π_c and Π_b, respectively.

Case 5: In this case, the peer $\pi_1 \in \Pi_1$ having the link to Π_c reduces its cost by replacing this link with a link to a peer in Π_b.

Case 6: Finally, this case is similar to Case 4: $\pi_1 \in \Pi_1$ with the link to Π_b has an incentive to remove its link to Π_c

These cases highlight how the system is ultimately trapped in an infinite loop of strategy changes, without ever converging to a stable situation. There is always at least one peer which can reduce its cost by changing its strategy. For instance, the following sequence of topology changes could repeat forever (cf Figure 9.2): $1 \rightsquigarrow 3 \rightsquigarrow 4 \rightsquigarrow 2 \rightsquigarrow 1 \rightsquigarrow 3 \ldots$ In other words, selfish peers will not achieve a stable network topology.

9.4 Complexity of Nash Equilibria

It remains to answer the question whether for a given p2p network, it can be determined if it will eventually converge to a stable state or not. In the following, we show that it is **NP**-hard to decide whether there exists a pure Nash equilibrium. This result establishes the *complexity of stability* in unstructured p2p networks, showing that in general, it is computationally infeasible to determine whether a peer-to-peer network consisting of selfish peers can stabilize or not.

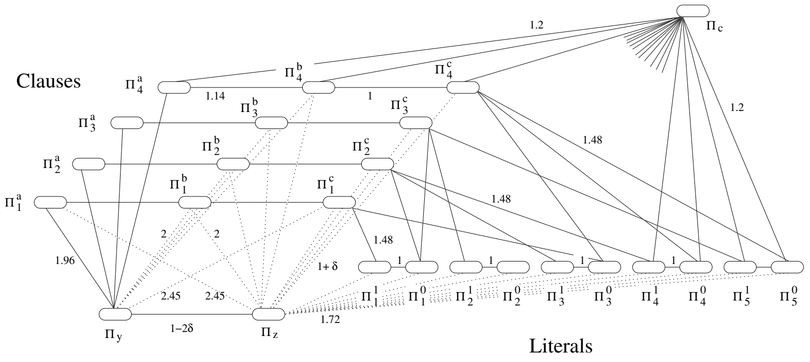


Figure 9.3: The graph G_I for instance $I = (x_1 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3 \vee x_4) \wedge (\overline{x_1} \vee x_2 \vee x_5) \wedge (\overline{x_3} \vee \overline{x_4} \vee \overline{x_5})$. Each cluster contains k peers with pairwise distance ϵ. δ is an arbitrary constant such that $\delta > \epsilon > 0$.

Theorem 9.6. *Regardless of the magnitude of α, determining whether a given p2p network represented by a metric space $\mathcal{M}$ has a pure Nash equilibrium (and can therefore stabilize) is* **NP***-hard.*

The proof being rather technical, we first describe its main intuition. The proof is based on a reduction from an **NP**-complete form of the Boolean satisfiability problem SAT which is restricted to instances with 2 or 3 variables per clause and at most 3 occurrences per variable [233]. For any α a multiple of 0.6, i.e., $\alpha_k = 0.6k$ for an integer $k > 0$, we give a polynomial time construction of a metric space $\mathcal{M}_I^k$ from an instance I of SAT, such that the following holds: there exists a pure Nash equilibrium in $\mathcal{M}_I^k$ if and only if I is satisfiable.

The reduction is illustrated in Figure 9.3, each rectangular box representing a cluster of k peers. Assume that the SAT instance is given in standard conjunctive normal form (CNF). For each clause C_j, we employ a gadget of three *clause-clusters* Π_j^a, Π_j^b, and Π_j^c. For every variable x_i, the two *literal-clusters* Π_i^0 and Π_i^1 represent the negative and positive literal of the variable, respectively. Finally, the construction's peer set is completed with three special clusters Π_c, Π_y, and Π_z. The pairwise distances between two peers in $\mathcal{M}_I^k$ are determined by the graph G_I^k shown in Figure 9.3 (a formal definition appears in Chapter 9.4.1). Two nodes within the same cluster have a distance of ϵ, for some arbitrarily small $\epsilon < (k(2n + 3m + 3))^{-2}$, where m and n denote the number of clauses and variables in I, respectively. An edge of G_I^k describes the *cluster-distance* between two clusters: the mutual distance between *every pair* of two peers $\pi_i \in \Pi_i$ and $\pi_j \in \Pi_j$ in neighboring clusters Π_i and Π_j with cluster-distance X is $d(\pi_i, \pi_j) = X$. All other distances are determined by the length of the shortest path between the peers in G_I^k, that is, $\mathcal{M}_I^k$ corresponds to the *shortest path metric* induced by G_I^k. Note that while $\mathcal{M}_I^k$ cannot be embedded in the Euclidean space, it still forms a valid metric space, i.e., it fulfils symmetry and triangle inequality.

Consider an arbitrary clause C_j. Its clause-clusters Π_j^a, Π_j^b, and Π_j^c in combination with the two special clusters Π_y and Π_z form an instance similar to I_k as used in the discussion of Theorem 9.5 (cf Figure 9.1). Hence, intuitively, when considering such a clause-gadget by itself, it does not have a pure Nash equilibrium. In order to make a clause-gadget stable, however, literal clusters may be used. For this purpose, the cluster-distance between each pair of corresponding literals is 1 and peers in Π_z have a distance of 1.72 to all literal-peers. Furthermore, the distance between a clause-cluster Π_j^c and a literal-cluster depends on whether the corresponding literal appears in the clause. Specifically, if the positive literal x_i appears in clause C_j, $x_i \in C_j$, the distance between Π_i^1 and Π_j^c is small, i.e., only 1.48. Similarly, if $\overline{x_i} \in C_j$, then $d(\Pi_i^0, \Pi_j^c) = 1.48$. And finally, if neither literal is in C_j, then there exists no short connection between the clusters, and the shortest distance between peers in these clusters is via Π_c.

The proof comprises two ingredients. First, we prove that if the underlying SAT instance I is *not satisfiable*, then there exists no Nash equilibrium. Towards this end, we show that in any Nash equilibrium two "neighboring" clusters (clusters connected by a short link in G_I^k, such as two clause-clusters in the same clause, a literal-cluster Π_i^1 to a clause-cluster Π_j^c if $x_i \in C_j$, or Π_c to all clause-clusters and literal-clusters, ...) always establish links in both directions between them. Between such close-by clusters, there are always exactly two links, one in each direction. Furthermore, for every variable x_i, there is exactly one peer $\pi_z \in \Pi_z$ that establishes a link to exactly either Π_i^1 or Π_i^0 (but not both!), while no other peer in Π_z links to these clusters.

From these lemmas, it then follows that because I is not satisfiable, there must exist a clause C_{j^*} for which the path from $\pi_z \in \Pi_z$ to peers in $\Pi_{j^*}^c$ via any literal-peer has a length of at least $d(\Pi_z, \Pi_i^\mu) + d(\Pi_i^\mu, \Pi_i^{1-\mu}) + d(\Pi_i^{1-\mu}, \Pi_{j^*}^c) = 4.2$, for $\mu \in \{0, 1\}$. This path being long, it follows that it is worthwhile for π_z to build an additional link directly to some peer in $\Pi_{j^*}^c$ or even in $\Pi_{j^*}^b$ instead. Based on these observations, we show that the subset of $\mathcal{M}_I^k$ induced by peers in Π_y, Π_z, and the clause-peers of C_{j^*} behaves similarly as in instance I_k of Figure 9.1. That is, peers in Π_y and Π_z continue to change their respective strategies forever, thus preventing the system from stabilizing.

On the other hand, if the SAT instance I has a *satisfying assignment* A_I, we explicitly construct a set of pure strategies that constitute a Nash equilibrium. In this strategy vector, one peer in Π_z builds a direct link to a peer in Π_i^1 if x_i is set to true in A_I and to a peer in Π_i^0 otherwise. Since A_I is a satisfying assignment, there must exist a path from Π_z via a single literal-cluster (i.e., without the additional detour of going from one literal-cluster to the other) to peers in every cluster Π_j^c. This path can be shown to have length at most $k\epsilon + d(\Pi_z, \Pi_i^\mu) + k\epsilon + d(\Pi_i^\mu, \Pi_j^c) + k\epsilon = 3.2 + 3k\epsilon$ from Π_z via a literal-cluster to peers in every cluster Π_j^c. It follows that in any satisfied clause C_j, the achievable reduction in stretch costs at a peer in Π_z when connecting directly to clusters Π_j^b or Π_j^c is significantly smaller than in an unsatisfied clause. Specifically, it can be shown that peers in Π_y and Π_z are in a *stable* situation if one peer $\pi_y \in \Pi_y$ connects to Π_j^a and Π_j^b of every clause C_j, and no peer in Π_z directly builds a link to any clause-peer. Since A_I is a satisfying assignment, peers in Π_y and Π_z are stable relative to *all* clauses in the SAT instance.

Furthermore, we also prove that in our strategy vector, no other peer in the network (i.e., peers in Π_c, Π_j^a, Π_j^b, Π_j^c, Π_i^1, or Π_i^0) has an incentive to deviate from its strategy. For this final ingredient of the proof, the existence of cluster Π_c is essential, because it ensures that all helper peers are mutually connected by optimal paths.

All in all, the p2p network induced by the metric space $\mathcal{M}_I^k$ has a pure Nash equilibrium if and only if the underlying SAT instance I is satisfiable. Hence, determining whether a given p2p network can stabilize is **NP**-hard. Chapter 9.4.1 defines the construction of G_I^k (and consequently $\mathcal{M}_I^k$) from the SAT instance I. In Chapters 9.4.2 and 9.4.3, we will show that there exists a Nash equilibrium in $\mathcal{M}_I^k$ if and only if I is satisfiable. Theorem 9.6 then follows from Lemmas 9.17 and 9.19, as well as the **NP**-hardness of SAT. The following theorem and proof is due to Tovey [233]:

Theorem 9.7. *Boolean satisfiability is* **NP***-hard when restricted to instances with 2 or 3 variables per clause and at most 3 occurrences per variable.*

Proof. Consider a general 3-SAT instance. For each variable x which appears in more than three clauses perform the following procedure: suppose x appears in k clauses. Create k new variables $x_1, ..., x_k$ and replace the i^{th} occurrence of x with x_i, $i = 1, ..., k$. Append the clause $\{x_i \vee \overline{x}_{i+1}\}$ for $i = 1, ..., k-1$ and the clause $\{x_k \vee \overline{x}_1\}$. In the new instance the clause $\{x_i \vee \overline{x}_{i+1}\}$ implies that if x_i is false, x_{i+1} must be false as well. The cyclic structure of the clauses therefore forces the x_i to have the same assignment, and hence the new instance is satisfiable if the original one is. As the transformation requires polynomial time, and as 3-SAT is **NP**-hard [68], the claim follows. □

9.4.1 The Construction of $\mathcal{M}_\mathbf{I}^\mathbf{k}$

Let I be an instance of SAT expressed in conjunctive normal form (CNF), in which each clause contains 2 or 3 variables. Without loss of generality, we can assume that each variable in I appears in at most 3 clauses [233]. Furthermore, we can restrict our attention to those instances of SAT in which each variable appears both as a positive and a negative literal at least once, because otherwise, assigning a feasible value to this variable is trivial. The set of clauses and variables of I is denoted by $\mathcal{C}$ and $\mathcal{X}$, respectively. Further, we write $m = |\mathcal{C}|$ and $n = |\mathcal{X}|$. Given I, we construct a graph $G_I^k = (V_I, E_I)$ in which each node represents a peer of the underlying network. Nodes are grouped into clusters of k peers and each cluster is illustrated as a rectangular box in Figure 9.3. Within each cluster, the pairwise distance between two peers is $\epsilon < (k(2n + 3m + 3))^{-2}$, and the distance between two peers in neighboring clusters is given by the *cluster-distance* $d(\Pi_i, \Pi_j)$

illustrated in Figure 9.3. The p2p network is then characterized by $\mathcal{M}_I^k$, which is induced by the *shortest path metric* of G_I^k, i.e., the distance between two peers corresponds to the length of the shortest path in G_I^k.

In more detail, G_I^k is defined as follows. The node-set V_I consists of three clusters of peers per clause $C_j \in \mathcal{C}$, denoted as *clause-clusters* Π_j^a, Π_j^b, and Π_j^c. Also, we add a pair of *literal-clusters* Π_i^0 and Π_i^1 for each of the n variables, with Π_i^1 representing the positive literal x_i, and Π_i^0 representing the negative literal $\overline{x_i}$. The set of clause-peers and literal-peers is denoted by C_P and L_P, respectively. Finally, there are three additional special clusters Π_c, Π_z, and Π_y. Call the union of Π_c and all clusters in C_P and L_P *top-layer clusters.* Peers in top-layer clusters are *top-layer peers.* The total number of peers N in the network $\mathcal{M}_I^k$ is therefore $N = k(2n + 3m + 3)$. Notice that $N \cdot \epsilon$ is smaller than $(k(2n + 3m + 3))^{-1}$.

The pairwise distances between the peers in different clusters—as illustrated in Figure 9.3—are as follows. Let δ be an arbitrarily small constant with $\delta > 10k\epsilon$, and $\mu \in \{0, 1\}$. For all $\pi_c \in \Pi_c$ and $\pi_w \in C_P \cup L_P$, it holds that $d(\pi_c, \pi_w) = 1.2$. For every $C_j \in \mathcal{C}$, the following distances apply.

$$\begin{array}{ll}
\forall \pi_y \in \Pi_y, \forall \pi_j^a \in \Pi_j^a : & d(\pi_y, \pi_j^a) = 1.96 \\
\forall \pi_y \in \Pi_y, \forall \pi_j^c \in \Pi_j^c : & d(\pi_y, \pi_j^c) = 2.45 \\
\forall \pi_z \in \Pi_z, \forall \pi_j^b \in \Pi_j^b : & d(\pi_z, \pi_j^b) = 2 \\
\forall \pi_j^a \in \Pi_j^a, \forall \pi_j^b \in \Pi_j^b : & d(\pi_j^a, \pi_j^b) = 1.14
\end{array}$$

$$\begin{array}{ll}
\forall \pi_y \in \Pi_y, \forall \pi_j^b \in \Pi_j^b : & d(\pi_y, \pi_j^b) = 2 \\
\forall \pi_z \in \Pi_z, \forall \pi_j^a \in \Pi_j^a : & d(\pi_z, \pi_j^a) = 2.45 \\
\forall \pi_z \in \Pi_z, \forall \pi_j^c \in \Pi_j^c : & d(\pi_z, \pi_j^c) = 2 + \delta \\
\forall \pi_j^b \in \Pi_j^b, \forall \pi_j^c \in \Pi_j^c : & d(\pi_j^b, \pi_j^c) = 1
\end{array}$$

For every variable $x_i \in \mathcal{X}$, it holds that

$$\begin{array}{lcl}
\forall \pi_i^0 \in \Pi_i^0, \forall \pi_i^1 \in \Pi_i^1 & : & d(\pi_i^0, \pi_i^1) = 1 \\
\forall \pi_z \in \Pi_z, \forall \pi_i^\mu \in \Pi_i^\mu & : & d(\pi_z, \pi_i^0) = d(\pi_z, \pi_i^1) = 1.72.
\end{array}$$

.
Furthermore,

$$\begin{array}{lcl}
\forall C_j \in \mathcal{C}, x_i \in C_j \forall \pi_i^1 \in \Pi_i^1, \forall \pi_j^c \in \Pi_j^c & : & d(\pi_i^1, \pi_j^c) = 1.48 \\
\forall C_j \in \mathcal{C}, \overline{x_i} \in C_j \forall \pi_i^0 \in \Pi_i^0, \forall \pi_j^c \in \Pi_j^c & : & d(\pi_i^0, \pi_j^c) = 1.48.
\end{array}$$

Finally, the distance between any two peers $\pi_y \in \Pi_y$ and $\pi_z \in \Pi_z$ is $d(\pi_y, \pi_z) = 1 - 2\delta$. All distances not explicitly defined follow from the shortest path metric induced by the above definitions.

Intuitively, the idea of the construction is the following. Each clause $C_j \in \mathcal{C}$ is represented by a gadget consisting of the two clusters Π_y, Π_z, as well as the clause-clusters Π_j^a, Π_j^b, and Π_j^c. By itself, each such gadget is reminiscent of the construction shown in Figure 9.1. Specifically, this implies that the sub-network induced by each such clause-gadget does not have a pure Nash equilibrium when considered independently from the rest of the network.

In order to render a clause-gadget stable, literal-peers can be used. In particular, it can be shown that for $\mu \in \{0,1\}$, the peers in every literal-cluster Π_i^μ construct links to those (at most two) clause-clusters Π_j^c in whose clause the literal occurs. Based on this and other structural properties of Nash equilibria in $\mathcal{M}_I^k$, it can further be shown that in a Nash equilibrium, there is exactly one link from cluster Π_z to each variable $x_i \in \mathcal{X}$, i.e., one peer in Π_z connects to a peer in either Π_i^0 or Π_i^1 for all $x_i \in \mathcal{X}$.

Consider a clause C_j. If there is a peer $\pi_z \in \Pi_z$ that connects to at least one literal-cluster that is directly connected to Π_j^c, the length of the path from π_z to peers in Π_j^c via this literal-cluster is at most $k\epsilon + d(\Pi_z, \Pi_i^\mu) + k\epsilon + d(\Pi_i^\mu, \Pi_j^c) + k\epsilon = 3.2 + 3k\epsilon$. In this case, the detour from π_z to Π_j^c via some "satisfying" literal-cluster Π_i^μ—while being suboptimal compared to the direct connection—is relatively small. Specifically, it is small enough to ensure that no peer in Π_z has an incentive to construct an additional direct link to Π_j^b or Π_j^c. Once peers in Π_z have no further need to establish direct links to a clause-peer of C_j, the best possible strategy of peers in Π_y becomes fixed, too. In other words, this satisfying literal helps in *stabilizing* the clause-gadget.

Conversely, if there is a clause C_j for which no peer in Π_z connects to a satisfying literal-cluster, there exists no efficient detour. Specifically, the length of the path from $\pi_z \in \Pi_z$ to $\pi_j^c \in \Pi_j^c$ via a literal-cluster is at least 4.2, including the distance between the positive and negative literal-cluster of the variable. The increased length of the detour renders the resulting stretch from Π_z to Π_j^c too large, and it becomes worthwhile for $\pi_z \in \Pi_z$ to construct direct links to Π_j^c, and even to Π_j^b. That is, in a sense, the network induced by the unsatisfied clause C_j becomes independent of the remainder of the network and therefore does not stabilize.

Finally, the special cluster Π_c ensures that the shortest path in G_I^k (and hence the distance in $\mathcal{M}_I^k$) between two top-layer peers is small. In fact, it can be shown that there are links in both directions from every top-layer cluster to Π_c. This implies that all top-layer clusters are connected to one another almost optimally (i.e., with low stretch) in every Nash equilibrium, thus facilitating the proof that such an equilibrium exists in case I is satisfiable. We end the section with a series of lemmas that capture structural properties of $\mathcal{M}_I^k$.

Lemma 9.8. *Consider two peers π_g and π_g' in an arbitrary cluster Π_g. In a Nash equilibrium, there exists a path from π_g to π_g' of length at most $k\epsilon$.*

Proof. Because the distance between π_g and π_g' is ϵ, it is easy to see that the shortest path between these two nodes must be located entirely in Π_g. Because the distance between each pair of peers in a cluster is ϵ and there are k peers in the cluster, the claim follows. □

Lemma 9.9. *Consider two arbitrary clusters Π_g and Π_h. In a Nash equilibrium, there is at most one peer $\pi_g \in \Pi_g$ that has a link to a peer in Π_h.*

Proof. Assume for contradiction that there are two nodes π_g and π_g' that maintain links to peers in Π_h. Then, π_g' can reduce its cost by dropping its link. Doing so, the stretches to each peer in the network can increase by at most $2k\epsilon$. By the definition of ϵ, it holds that $2Nk\epsilon < \alpha$ and hence, dropping the link is worthwhile. □

Based on these two lemmas, we can go on to prove more elaborate properties.

Lemma 9.10. *Let Π_g and Π_h be two clusters with cluster distance at most $d(\Pi_g, \Pi_h) \leq 1.48$. In any Nash equilibrium, there is exactly one peer $\pi_g \in \Pi_g$ that has a link to a peer in Π_h.*

Proof. By Lemma 9.9, there cannot be more than one peer in Π_g having a link to Π_h. It therefore remains to show that at least one link exists. We divide the proof in two parts and begin by showing that the claim holds for all pairs of clusters with cluster distance $d(\Pi_g, \Pi_h) \leq 1.2$. In a second step, we prove the claim for pairs of clusters with cluster distance $d(\Pi_g, \Pi_h) = 1.48$, which suffices because there are no cluster distances between 1.2 and 1.48 in G_I^k.

Consider any two clusters in the network $\mathcal{M}_I^k$ with cluster distance at most 1.2. It follows from the construction of G_I^k that the shortest path between peers in these clusters via a third cluster has a length of at least 2.2 (e.g., from Π_i^0 via Π_i^1 to Π_c). In other words, if there is no direct connection between the two clusters, π_g has a stretch of at least $2.2/1.2$ to each peer in Π_h. Because $\frac{2.2k}{1.2} > \alpha + k(1+2k\epsilon)$, it is beneficial for π_g to establish a direct link to the other cluster.

For the second part of the proof, consider pairs of clusters with cluster distance $d(\Pi_g, \Pi_h) = 1.48$. Specifically, we need to show the existence of a link in each direction between clusters Π_j^c and Π_i^1, if $x_i \in C_j$, or between Π_j^c and Π_i^0, if $\overline{x_i} \in C_j$. The shortest indirect connection between two such clusters has a length of at least 2.4 (via cluster Π_c) and hence, the cumulated stretch to all peers in the respective cluster without a direct link is $\frac{2.4k}{1.48} > \alpha + k(1 + 2k\epsilon)$. That is, peers in both clusters decrease their cost by paying for this direct link. □

Lemma 9.10 implies that within a clause, neighboring clause-clusters (i.e., $\Pi_j^a \leftrightarrow \Pi_j^b$ and $\Pi_j^b \leftrightarrow \Pi_j^c$, respectively) are connected in both directions in any Nash equilibrium. The same holds for corresponding literal-cluster Π_i^1 and Π_i^0, as well as for a literal-cluster Π_i^1 (or Π_i^0) and a Π_j^c if $x_i \in C_j$ (or $\overline{x_i} \in C_j$). Also, there are links in both directions from any top-layer (clause or literal) cluster to Π_c and vice versa. All in all, this implies that in a Nash equilibrium, every pair of top-layer peers is connected almost optimally, i.e., with stretch of less than $1 + 2k\epsilon$. The value ϵ being smaller than $(k(m + n + 3))^{-2}$, this stretch is virtually as good as 1. Finally, there are also links between Π_y and

Π_z in any Nash equilibrium. In the sequel of the proof, we use the fact that these "short" links are available in any Nash equilibrium without particular mention.

Lemma 9.11. *In a Nash equilibrium, there is exactly one peer $\pi_y \in \Pi_y$ that has a link to a peer in Π_j^a, for all $C_j \in \mathcal{C}$, and vice versa.*

Proof. Consider a specific Π_j^a. If there exists no direct link from Π_y to Π_j^a, the stretch of a peer $\pi_y \in \Pi_y$ to each peer in Π_j^a is at least $\frac{3.14}{1.96}$. Because for small enough ϵ, we have $\frac{3.14k}{1.96} > \alpha + k(1+2k\epsilon)$, it is always worthwhile for some π_y to build an additional link to Π_j^a. Clearly, the argument also holds for the opposite direction. □

Lemma 9.12. *Assume that there is a link between Π_z and at least one literal-cluster of every variable $x_i \in \mathcal{X}$ and that there is a link between Π_y and Π_j^a, for all $C_j \in \mathcal{C}$. Assume further that there are links in both directions between clusters with cluster distance at most 1.48. Finally, assume that all peers are connected within their cluster with a path of length at most $k\epsilon$. It holds for all j that the shortest path from a peer $\pi_y \in \Pi_y$ to a peer in $V \setminus (\Pi_j^a \cup \Pi_j^b \cup \Pi_j^c)$ is not via Π_j^a, Π_j^b, or Π_j^c, even when directly connecting to such a cluster. The same holds for $\pi_z \in \Pi_z$.*

Proof. Recall that by assumption there exists a link from Π_y to Π_j^a (for every $C_j \in \mathcal{C}$) and Π_z. Hence, connecting to Π_j^b or Π_j^c clearly cannot reduce the stretch to peers in Π_z, Π_c, and any $\Pi_{j'}^a$, $j \neq j'$. Furthermore, the distance in the topology to any clause-peer in $\Pi_{j'}^b$ and $\Pi_{j'}^c$ via $\Pi_{j'}^a$ is at most $3.1 + 3k\epsilon$ and $4.1 + 4k\epsilon$, respectively, which is strictly smaller than $2 + 2 \cdot 1.2 = 4.4$, which is the shortest achievable distance via Π_j^b or Π_j^c. Finally, the path from $\pi_y \in \Pi_y$ to any literal-peer in Π_i^μ has a length of at most $3.72 - 2\delta + 3k\epsilon$. This is because there exists a link between Π_y and Π_z, and between Π_i^0 and Π_i^1, and because there is a link from Π_z to either Π_i^0 or Π_i^1. On the other hand, the path from $\pi_y \in \Pi_y$ to a literal-peer via Π_j^b or Π_j^c has a length of at least $2.45 + 1.48 = 3.93$. Similar arguments show that the same holds for $\pi_z \in \Pi_z$. □

9.4.2 Satisfiable Instances

In this section, we show that if I has a satisfying assignment A_I, then there exists a Nash equilibrium in $\mathcal{M}_I^k$. For this purpose, we explicitly construct a set of strategies s, which we prove to constitute a Nash equilibrium. Let $A_I(x_i)$ be the assignment of x_i in A_I, i.e.,

$$A_I(x_i) := \begin{cases} 1 \text{ , } x_i \text{ is set to 1 in } A_I \\ 0 \text{ , } x_i \text{ is set to 0 in } A_I. \end{cases} \tag{9.1}$$

Furthermore, we define in every cluster Π_g a single *leader peer*, which we denote by $\hat{\pi}_g$. The role of this leader-peer is to construct all inter-cluster links going from this cluster to peers located in other clusters. The strategy

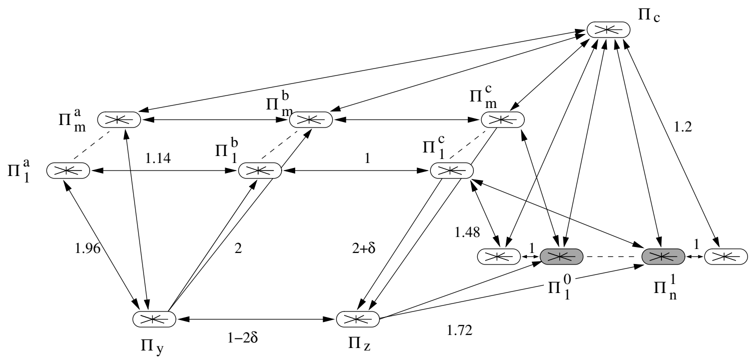


Figure 9.4: An example instance G_I^k with the topology resulting from strategy s. Within each cluster, the peers are connected as a star. Directed arrows between clusters indicate inter-cluster links between cluster-leaders. Cluster-leader $\hat{\pi}_z$ connects to those leaders of literal-peers that appear in the satisfying assignment A_I. In the example, A_I sets $x_1 = 0$ and $x_n = 1$.

of the remaining *non-leader peers* $\check{\pi}_g \in \Pi_g \setminus \{\hat{\pi}_g\}$ is to connect to the unique leader peer within their cluster. Formally, the strategy s_g for a non-leader peer $\check{\pi}_g \in \Pi_g \setminus \{\hat{\pi}_g\}$ is $s_g := \{\hat{\pi}_g\}$. For each leader-peer, we define the set of strategies s as follows:

$$\begin{aligned}
s_y &:= \Pi_y \cup \{\hat{\pi}_z\} \cup \textstyle\bigcup_{C_j \in \mathcal{C}} \{\hat{\pi}^a_j, \hat{\pi}^b_j\}\\
s_c &:= \Pi_c \cup \textstyle\bigcup_{x_i \in \mathcal{X}} \{\hat{\pi}^0_i \cup \hat{\pi}^1_i\} \cup \bigcup_{C_j \in \mathcal{C}} \{\hat{\pi}^a_j \cup \hat{\pi}^b_j \cup \hat{\pi}^c_j\}\\
s^c_j &:= \Pi^c_j \cup \{\hat{\pi}_c, \hat{\pi}_z, \hat{\pi}^b_j\} \cup \textstyle\bigcup_{x^\mu_i \in C_j} \{\hat{\pi}^\mu_i\}, \quad \forall C_j \in \mathcal{C}\\
s^\mu_i &:= \Pi^\mu_i \cup \{\hat{\pi}_c, \hat{\pi}_z, \hat{\pi}^{1-\mu}_i\} \cup \textstyle\bigcup_{x^\mu_i \in C_j} \{\hat{\pi}^c_j\}, \ \forall x_i \in \mathcal{X}
\end{aligned}$$

$$\begin{aligned}
s_z &:= \Pi_z \cup \{\hat{\pi}_y\} \cup \textstyle\bigcup_{x_i \in \mathcal{X}} \{\hat{\pi}^{A_I(x_i)}_i\}\\
s^a_j &:= \Pi^a_j \cup \{\hat{\pi}_c, \pi_y, \hat{\pi}^b_j\}, \quad \forall C_j \in \mathcal{C}\\
s^b_j &:= \Pi^b_j \cup \{\hat{\pi}_c, \hat{\pi}^a_j, \hat{\pi}^c_j\}, \ \forall C_j \in \mathcal{C}
\end{aligned}$$

Strategy s is illustrated in Figure 9.4. Our goal is to show that s constitutes a Nash equilibrium for A_I. The topology resulting from strategy s contains all "short" links, i.e., links between cluster leaders of clusters that have a distance of at most 1.48 (cf Lemma 9.10). Additionally, peer $\hat{\pi}_y$ builds links to clause-cluster leaders $\hat{\pi}^a_j$ and $\hat{\pi}^b_j$ for all $C_j \in \mathcal{C}$. On the other hand, leaders $\hat{\pi}^a_j$ and $\hat{\pi}^c_j$ have a link to $\hat{\pi}_y$ and $\hat{\pi}_z$, respectively. Most importantly, however, for every variable $x_i \in \mathcal{X}$, leader-peer $\hat{\pi}_z$ maintains a link to the literal-peers $\hat{\pi}^{A_I(x_i)}_i$ that are used in the satisfying assignment. Note that because in s, peer $\hat{\pi}_z$ has exactly one connection to a literal-peer of every variable, we can apply Lemma 9.12. That is, no peer in clusters Π_y and Π_z can reduce its stretch to any peer $V \setminus (\Pi_j^a \cup \Pi_j^b \cup \Pi_j^c)$ by connecting to one of the clause-peers of clause C_j. Finally, note that non-leaders are directly connected to their cluster leader, and cluster leaders maintain direct links to each peer in their cluster.

The next three lemmas prove that no peer has an incentive to single-handedly deviate from strategy s. In the proofs, we use the notation $\Delta_i(\psi)$ to denote the change in cost when peer π_i changes its strategy according to action ψ, ψ being clear from the context. Specifically, if $\Delta_i(\psi) \geq 0$, peer π_i has no incentive to perform action ψ because doing so would increase its cost.

We begin with a lemma that shows that no peer can unilaterally benefit from changing its links within its own cluster.

Lemma 9.13. *In s, no peer in an arbitrary cluster Π_g has an incentive to change its strategy within the cluster, i.e., to add, replace, or remove links to peers in Π_g.*

Proof. The cluster leader $\hat{\pi}_g$ cannot remove any link because the topology would become disconnected without it. Next, consider a non-leader $\check{\pi}_g$. If $\check{\pi}_g$ removes its link to the cluster-leader, it disconnects itself from the network. Adding one or more new link to a non-leader costs α per link, while the resulting stretch reduction per link is $\frac{2\epsilon}{\epsilon} - 1 = 1$ only. Finally, replacing the link to the leader with a link to another non-leader strictly increases the stretch to all but one peer in the network and therefore cannot be beneficial. □

Based on Lemma 9.13, we can regard the topology within each cluster in s as fixed. It remains to show that no peer has an incentive to add, remove, or replace its inter-cluster links. As shown next, peers in Π_y cannot unilaterally reduce their costs in s.

Lemma 9.14. *No peer in Π_y has an incentive to change its strategy, given that all other peers follow strategy s.*

Proof. By Lemma 9.13, no peer $\pi_y \in \Pi_y$ has an incentive to change its intra-cluster links. Furthermore, $\hat{\pi}_y$ does not benefit from switching its link from a leader peer to a non-leader peer, because this would only decrease the stretch to that particular peer, while increasing the stretch to all other peers (at least) in this cluster. It follows from Lemmas 9.10 and 9.11 that $\hat{\pi}_y$ must keep its links to $\hat{\pi}_j^a$ and $\hat{\pi}_z$. We now show that no peer in Π_y can reduce this cost by deviating from its strategy in any other way.
Case 1: *Some $\check{\pi}_y$ or $\hat{\pi}_y$ adds one or more additional links*: in the topology resulting from s, every peer in Π_y is connected with stretch at most $1 + 2\epsilon$ with all peers except from peers in Π_j^c (for all $C_j \in \mathcal{C}$) and peers in those literal-clusters to which $\hat{\pi}_z$ does not have a direct connection. With any additional link, a peer in Π_y can reduce its stretch to peers in exactly one

of these clusters only. Hence, every additional link would increase the peer's cost: $\Delta_y(+) \geq -\frac{k(4.72+\epsilon)}{3.72} + \alpha + k > 0$.

Observe that because non-leader peers $\check{\pi}_y \in \Pi_y$ do not have inter-cluster links, Case 1 in combination with Lemma 9.13 implies that no $\check{\pi}_y$ can benefit from changing its strategy.

Case 2: *$\hat{\pi}_y$ changes its link from $\hat{\pi}_j^b$ to $\hat{\pi}_j^c$*: while the stretch to peers in Π_j^c is reduced, the stretch to peers in Π_j^b increases. The relative cost difference is $\Delta_y(\hat{\pi}_j^b \rightarrow \hat{\pi}_j^c) \geq -\frac{k(3+\epsilon)}{2.45} + \frac{(1.96+1.14)k}{2} > 0$.

Case 3: *$\hat{\pi}_y$ removes its link from $\hat{\pi}_j^b$*: by removing such a link, $\hat{\pi}_y$ can save the link's cost α. On the other hand, the stretch to both Π_j^b and Π_j^c increase. Specifically, the shortest connection to peers in these clusters is now via $\hat{\pi}_j^a$ and $\hat{\pi}_j^b$, i.e., $\Delta_y(-\hat{\pi}_j^b) \geq -\alpha - k(1+\epsilon) - \frac{k(3+\epsilon)}{2.45} + \frac{(1.96+1.14)k}{2} + \frac{(1.96+1.14+1)k}{2.45} > 0$.

The only other thing that could potentially lead to an advantage for $\hat{\pi}_y$ is to replace a link $\hat{\pi}_j^b$ by some leader peer in Π_i^μ to which $\hat{\pi}_z$ is not connected, formally $\mu \neq A_I(x_i)$. Doing so clearly increases the stretch to peers in Π_j^b and Π_j^c, but like in Case 3, the shortest connection between $\hat{\pi}_y$ to peers in Π_j^c is via $\hat{\pi}_j^a$ and $\hat{\pi}_j^b$. In particular, this path has a length of at most $4.1+\epsilon$, whereas the shortest path via a literal-cluster has a length of at least $1-2\delta+1.72+1.48 = 4.2-2\delta$, which is larger. Hence, replacing one or more links to $\hat{\pi}_j^b$ by links to literal-peers reduces to Cases 1 and 3, respectively, and therefore cannot be worthwhile. Finally, no combination of the above cases can reduce the cost of any peer in Π_y either. □

Lemma 9.15. *No peer in Π_z has an incentive to change its strategy, given that all other peers follow strategy s.*

Proof. Again, we discuss the various cases and show that none of them is beneficial for a peer in Π_z. Recall that by Lemma 9.12, connecting to any clause-peer cannot improve the stretch to any other peer outside this clause. Furthermore, because A_I is a satisfying assignment, the topology of s contains a path of length at most $\epsilon + d(\Pi_z, \Pi_i^\mu) + d(\Pi_i^\mu, \Pi_j^c) + \epsilon = 3.2 + 2\epsilon$ between peers in Π_z and peers in Π_j^c, for *every* clause $C_j \in \mathcal{C}$. Consequently, connecting to a so far unconnected literal-peer cannot decrease the stretch to any clause-peer $\pi_j \in C_P$ in the system.

It follows from Lemma 9.13 that no peer $\pi_z \in \Pi_z$ has an incentive to change its intra-cluster links. Also, as shown in the proof of Lemma 9.14, no peer can benefit from connecting to a non-leader peer in the network, because this bears strictly higher costs than connecting to the corresponding leader peer of the same cluster. Hence, we only need to verify the cases in which peers in Π_z connect to leader peers.

In the following, we will discuss the various cases how peers in Π_z could improve their situation and derive that none of them is actually beneficial.

Case 1: *Some peer in Π_z adds an additional link to $\hat{\pi}_j^b$*: the reduction of the stretches to peers in Π_j^b and Π_j^c resulting from the additional link does not outweigh the link's cost. Specifically, we have $\Delta_z(+\hat{\pi}_j^b) \geq -\frac{k(3-2\delta+2\epsilon)}{2} + k -$

$\frac{k(3.2+2\epsilon)}{2+\delta} + \frac{3k}{2+\delta} + \alpha \geq k(4\delta + 2\delta^2) > 0$. Notice that in the second term, the stretch to each of the k peers in Π_j^b is at least 1, and in the third term, the distance $3.2 + 2\epsilon$ holds because A_I is a satisfying assignment.
Case 2: *Some peer in* Π_z *adds an additional link to* $\hat{\pi}_j^c$: again, the stretches to Π_j^b and Π_j^c are not reduced enough to render the additional link worthwhile. In fact, the stretch to peers in Π_j^b is not reduced by the addition of this link, nor is the stretch to any other peer in the network except from peers in Π_j^c (Lemma 9.12). It follows that $\Delta_z(+\hat{\pi}_j^c) \geq -\frac{k(3.2+2\epsilon)}{2+\delta} + k + \alpha = k(1.6\delta - 2\epsilon) > 0$.
Case 3: *Some peer in* Π_z *adds an additional link to* $\hat{\pi}_j^a$: clearly, this option is even worse than Cases 1 and 2.
Case 4: *Some peer in* Π_z *adds an additional link to* $\hat{\pi}_i^\mu$: adding a link to a literal-cluster that is not used in A_I reduces the stretch to peers in this cluster only, because there is already a short connection from Π_z to every Π_j^c through the literal-clusters $\Pi_i^{A_I(x_i)}$. Hence, $\Delta_z(+\hat{\pi}_i^\mu) \geq -\frac{k(2.72+2\epsilon)}{1.72} + k + \alpha > 0$.

Observe that because non-leader peers $\breve{\pi}_z \in \Pi_z$ do not have inter-cluster links, Cases 1 to 4 in combination with Lemma 9.13 implies that no $\breve{\pi}_z$ can benefit from changing its strategy.
Case 5: $\hat{\pi}_z$ *replaces some* $\hat{\pi}_i^{A_I(x_i)}$ *by* $\hat{\pi}_i^{1-A_I(x_i)}$: again, the new link to a previously unconnected literal-cluster cannot decrease the stretch to any clause-peer, because A_I is a satisfying assignment and $\hat{\pi}_z$ already had a path of length 3.2 to every $\hat{\pi}_j^c$ via some $\hat{\pi}_i^{A_I(x_i)}$. Furthermore, by a symmetry argument, the stretch cost gained by adding the link to $\hat{\pi}_i^{1-A_I(x_i)}$ is lost by removing the link to $\hat{\pi}_i^{A_I(x_i)}$. Thus, $\Delta_y(\hat{\pi}_i^{A_I(x_i)} \rightarrow \hat{\pi}_i^{1-A_I(x_i)}) \geq 0$.
Case 6: $\hat{\pi}_z$ *removes or replaces some* $\hat{\pi}_i^{A_I(x_i)}$: if $\hat{\pi}_z$ does not have a connection to any literal-cluster of a variable x_i, the resulting stretch to each peer in these two clusters is at least $\frac{3+\delta+1.48}{1.72}$. Because $\frac{k(4.48+\delta)}{1.72} > k(1 + 2\epsilon) + \alpha$, it follows that $\hat{\pi}_z$ must maintain at least one link to such a peer.

Any other possible strategy deviation can either be reduced to one of the above five cases or to Lemma 9.10. □

Having shown that peers in Π_y and Π_z have no incentive to deviate from s, we have to prove that no other peer can improve its situation either.

Lemma 9.16. *No top-layer peer can benefit from changing its strategy, given that all other peers follow* s.

Proof. First, by Lemma 9.13, it holds that no peer can improve its situation by adding, replacing, or removing a link within its cluster. Also, no peer can benefit from connecting to a non-leader, as opposed to the leader peer in the same cluster. Both claims can be proven with exactly the same argument as in the proof of Lemma 9.14.

It is important to observe that in s, all top-layer peers are almost optimally connected with each other, either via the central cluster Π_c or because their respective clusters are neighbors in the graph. More specifically, the stretch between any pair of top-layer peers in s is at most $1+2\epsilon$ (via the own

cluster leader, $\hat{\pi}_c$, and the other cluster leader). Besides removing the final 2ϵ from these small stretches, adding additional links can only help in reducing the stretches to peers in Π_y and Π_z. By Lemma 9.10, no link between cluster leaders whose clusters have a distance of less than 1.48 can be removed from s. Hence, the possible strategy deviations by other nodes is actually limited.
Peers in $\mathbf{\Pi_j^a}$: A peer $\hat{\pi}_j^a$'s link to $\hat{\pi}_y$ cannot be removed by Lemma 9.11. For every peer $\pi_j^a \in \Pi_j^a$, it further holds that building an additional link to $\hat{\pi}_z$ is too costly, $\Delta_j^a(+\hat{\pi}_z) \geq -\frac{k(2.96-2\delta+2\epsilon)}{2.45} + k + \alpha > 2kN\epsilon$. Hence, even if this additional link could reduce all other less than N stretches to top-level peers by the remaining 2ϵ, the cost of an additional link would still be too high.
Peers in $\mathbf{\Pi_j^b}$: Peer $\hat{\pi}_j^b$ does not have a link longer than 1.48 in s and hence, cannot remove any of them. We show that neither building a link to $\hat{\pi}_y$ nor to $\hat{\pi}_z$ decreases the cost of any peer in Π_j^b. In the first case, we have $\Delta_j^b(+\hat{\pi}_y) \geq -\frac{k(1.96+1.14+2\epsilon)}{2} + k - \frac{k(3+\delta+2\epsilon)}{2} + \frac{k(3-2\delta)}{2} + \alpha > 2kN\epsilon$. As for the second case, $\Delta_j^b(+\hat{\pi}_z) \geq -\frac{k(1.96+1.14+2\epsilon)}{2} + \frac{k(3-2\delta)}{2} - \frac{k(3+\delta+2\epsilon)}{2} + k + \alpha > 2kN\epsilon$. Clearly, building both links is even less worthwhile.
Peers in $\mathbf{\Pi_j^c}$: The potential strategy deviations that could decrease peer $\hat{\pi}_j^c$'s costs are to add a link to $\hat{\pi}_y$, to remove its link from $\hat{\pi}_z$, or to replace the link to $\hat{\pi}_z$ by a link to $\hat{\pi}_y$. However, none of these alterations are beneficial for $\hat{\pi}_j^c$ (or for any non-leader peer in Π_j^c in the case of link addition). First, it holds that $\Delta_j^c(+\hat{\pi}_y) \geq -\frac{k(3-\delta+2\epsilon)}{2.45} + k + \alpha > 2kN\epsilon$ and $\Delta_j^c(-\hat{\pi}_z) \geq -\alpha - k(1+2\epsilon) - \frac{k(3-\delta+2\epsilon)}{2.45} + \frac{3.2k}{2+\delta} + \frac{4.1k}{2.45} > 2kN\epsilon$. Also, switching the link from $\hat{\pi}_z$ to $\hat{\pi}_y$ is not helpful, $\Delta_j^c(\hat{\pi}_z \rightarrow \hat{\pi}_y) \geq \frac{3.2k}{2+\delta} - \frac{k(3-\delta+2\epsilon)}{2.45} > 2kN\epsilon$.
Peers in $\mathbf{\Pi_i^\mu}$: Each leader of a literal-cluster maintains a link to $\hat{\pi}_z$, and we show that they (as well as any non-leader peer in these clusters) do not have an incentive to change that strategy. It is clear that neither adding a link to $\hat{\pi}_y$ nor switching from $\hat{\pi}_z$ to $\hat{\pi}_y$ can be beneficial. In the first case, the stretch is reduced by at most 2ϵ by the additional link, which does not render the link cost α worthwhile. In the second case, the stretch is strictly increased. If $\hat{\pi}_i^\mu$ removes its link to $\hat{\pi}_z$ and connects via its neighboring literal-cluster, the stretches to both Π_y and Π_z increase. Particularly, we have $\Delta_i^\mu(-\hat{\pi}_z) \geq -\alpha - k(1+2\epsilon) + \frac{2.72k}{1.72} + \frac{k(3.72-2\delta)}{2.72-2\delta} > 2kN\epsilon$.
Peers in $\mathbf{\Pi_c}$: Finally, peers in Π_c are connected with stretch at most 2ϵ to all peers in the network. To top-clusters, the connection is via links shorter than 1.48. As for the remaining two clusters, it is connected to $\hat{\pi}_z$ via one of the literal-clusters and to $\hat{\pi}_y$ via some $\hat{\pi}_j^a$. By the definition of ϵ and α, it is clear that no peer in Π_c can improve its strategy. □

By combining Lemmas 9.14, 9.15, and 9.16, we know that no peer in the network has an incentive to change its strategy. Hence, s constitutes a pure Nash equilibrium.

Lemma 9.17. *If I is satisfiable, there exists a pure Nash equilibrium in $\mathcal{M}_I^k$.*

9.4.3 Non-satisfiable Instances

It remains to prove the other direction, that is, there exists no pure Nash equilibrium in the network if the underlying SAT instance I has no satisfying assignment. We proceed by defining structural properties that any Nash equilibrium must fulfil, and show that the intersection of all these properties is empty. Besides the basic properties derived in Chapter 9.4.1, an important characteristic of any Nash equilibrium is the fact that exactly one peer in Π_z connects to *exactly one* literal-peer (either in Π_i^0 or Π_i^1) for every variable $x_i \in \mathcal{X}$.

Lemma 9.18. *In any Nash equilibrium, exactly one peer in Π_z connects to either a peer $\pi_i^1 \in \Pi_i^1$ or $\pi_i^0 \in \Pi_i^0$, for every $x_i \in \mathcal{X}$.*

Proof. We have already shown in Lemma 9.15 (Case 6) that there must be a peer $\pi_z \in \Pi_z$ that has at least one link to a literal-peer of every variable. Furthermore, we know by Lemma 9.9 that no other peer in Π_z connects to the same cluster as π_z. Hence, we only need to show that in a Nash equilibrium no two peers in Π_z connect to both literal-clusters of the same variable.

Assume for the sake of contradiction that peers π_z and π_z' (potentially $\pi_z = \pi_z'$) maintain links to both Π_i^0 and Π_i^1 for some $x_i \in \mathcal{X}$. In this case, it would be worthwhile for one of the two peers to remove its link and replace it with a link to some peer in Π_j^c if this link does not already exist. By the definition of our special SAT instance and the construction of G_I, we know that of the two literal-clusters, one, say Π_i^μ, has clause-cluster Π_j^c at distance 1.48, and the other literal-cluster, say $\Pi_i^{1-\mu}$, has one or two such close-by clause-clusters. Let π_z' be the peer that connects to cluster Π_i^μ (otherwise, replace π_z with π_z' for the remainder of the proof).

Assume for the first case that the length of the shortest path from π_z' to this Π_j^c without the link via Π_i^μ is 3.2 or longer. In this case, the change in π_z''s costs when switching from its link to literal-cluster Π_i^μ that has only a single close-by clause-cluster Π_j^c directly to a peer in Π_j^c is $\Delta_z(\pi_i^\mu \to \pi_j^c) \leq +\frac{k(2.72+2k\epsilon)}{1.72} - \frac{3.2k}{2+\delta} + \frac{k(2+\delta+2k\epsilon)}{2+\delta} < 0$. If the length of the path from π_z to Π_j^c is strictly shorter than 3.2, then the link to Π_i^μ can simply be dropped, resulting in a gain of $\Delta_z(-\pi_i^\mu) \leq -\alpha - k + \frac{k(1.72+2k\epsilon)}{1.72} + \frac{k(2.72+2k\epsilon)}{1.72} < 0$. Hence, π_z' is always better off not connecting to a literal-cluster if π_z already connects to a literal-cluster. From this, the claim follows. □

Lemma 9.18 is an important ingredient for the remainder of the proof, because it gives us a one-to-one correspondence between the connections of Π_z to literal-clusters, and an assignment of variables in the SAT instance I. Also, note that when combining Lemma 9.18 with Lemma 9.12, it follows that in a Nash equilibrium, peers in Π_y and Π_z cannot reduce their stretch to any peer in $V \setminus \{\Pi_j^a \cup \Pi_j^b \cup \Pi_j^c\}$ by connecting to one of the clause-peers of clause C_j.

Lemma 9.19. *If I is non-satisfiable, there exists no pure Nash equilibrium in $\mathcal{M}_I^k$.*

Proof. By Lemma 9.18, exactly one peer in Π_z connects to either the positive or negative literal-cluster of every variable x_i. Because there exists no satisfying assignment, it follows that regardless of how Π_z is connected to the literal-clusters, there must exist at least one clause C_{j^*} that is "not satisfied". In the resulting topology, this means that the path from a peer in Π_z to a clause-peer in $\Pi^c_{j^*}$ of this unsatisfied clause via any literal-cluster must be of length at least $d(\Pi_z, \Pi^{\mu}_i) + d(\Pi^{\mu}_i, \Pi^{1-\mu}_i) + d(\Pi^{1-\mu}_i, \Pi^c_{j^*}) = 4.2$. Particularly, every such path must include the additional distance of 1 between x^1_i and x^0_i. In the sequel, we consider this *unsatisfied clause* C_{j^*} in more detail.

First, we show that in a Nash equilibrium, no peer $\pi_y \in \Pi_y$ establishes a link to $\Pi^c_{j^*}$. We distinguish two cases. In the first case, if some peer in Π_y already has a link to $\Pi^b_{j^*}$, then the cost reduction for π_y when omitting its link to $\Pi^c_{j^*}$ is $\Delta_y(-\pi^c_{j^*}) \leq -\alpha - k + \frac{k(3+2k\epsilon)}{2.45} < 0$. In the other case, the cost reduction when switching the link from $\Pi^c_{j^*}$ to a peer in $\Pi^b_{j^*}$ is at least $\Delta_y(\pi^c_{j^*} \rightarrow \pi^b_{j^*}) \leq -\frac{k(3-2\delta)}{2} + \frac{k(3+2k\epsilon)}{2.45} < 0$. That is, in either case it is beneficial for π_y not to connect directly to $\Pi^c_{j^*}$.

For the next step, we establish that in any Nash equilibrium, exactly one peer $\pi_z \in \Pi_z$ connects to either a peer in $\Pi^b_{j^*}$ or in $\Pi^c_{j^*}$. To see this, assume that no peer in Π_z establishes any links to peers in the two clusters. In this case (because there is no link from Π_y to $\Pi^c_{j^*}$, and because C_{j^*} is not satisfied), the sum of the stretches to peers in $\Pi^c_{j^*}$ is at least $\frac{k(4-2\delta)}{2+\delta} > k(1 + 2k\epsilon) + \alpha$. That is, $\pi_z \in \Pi_z$ can reduce its cost by connecting to $\pi^c_{j^*}$.

It remains to show that no peer in Π_z connects to $\Pi^a_{j^*}$, and particularly, that no two peers in Π_z simultaneously connect to both $\Pi^b_{j^*}$ or $\Pi^c_{j^*}$. Because there is at least one link from Π_z to either $\Pi^b_{j^*}$ or $\Pi^c_{j^*}$, it follows that a link to $\Pi^a_{j^*}$ can only reduce the stretch to peers in this particular cluster. However, the incurred cost exceeds the savings due to the reduced stretch, i.e., $\Delta_z(+\pi^a_{j^*}) = -\frac{k(2.96-2\delta+2k\epsilon)}{2.45} + \alpha + k > 0$. For the last case, assume that two peers π_z and π'_z (potentially the same) connect to both $\Pi^b_{j^*}$ and $\Pi^c_{j^*}$, respectively. Then, π'_z has an incentive to drop its link to $\Pi^c_{j^*}$: $\Delta_z(-\pi^c_{j^*}) = \frac{k(3+2k\epsilon)}{2+\delta} - k - \alpha < 0$. Hence, in any Nash equilibrium, there is exactly one link from Π_z to either $\Pi^b_{j^*}$ or $\Pi^c_{j^*}$, but not to both.

Studying the above rules, it can be observed that there remain only four possible sets of strategies for peers in Π_y and Π_z that could potentially result in a pure Nash equilibrium. The four cases can be distinguished by whether or not a peer in Π_y directly connects to $\Pi^b_{j^*}$, and by whether a peer in Π_z connects to $\Pi^b_{j^*}$ or $\Pi^c_{j^*}$.

Case 1: Some peer $\pi_z \in \Pi_z$ connects to $\pi^b_{j^*}$: in this case, some peer $\pi_y \in \Pi_y$ has an incentive to add a link to a peer in $\Pi^b_{j^*}$, because this significantly reduces its stretches to peers in $\Pi^b_{j^*}$ and $\Pi^c_{j^*}$. Specifically, π_y could reduce its cost by at least $\Delta_y(+\pi^b_{j^*}) \leq -\frac{k(3-2\delta)}{2} - \frac{k(4-2\delta)}{2.45} + \alpha + k(1+2k\epsilon) + \frac{k(3+2k\epsilon)}{2.45} < 0$.
Case 2: Peers $\pi_z \in \Pi_z$ and $\pi_y \in \Pi_y$ connect to $\Pi^b_{j^*}$: in this case, the peer π_z can profit from switching its link to a peer in $\Pi^c_{j^*}$. Specifically,

$\Delta_z(\pi^b_{j*} \rightarrow \pi^c_{j*}) \leq -\frac{3k}{2+\delta} + \frac{k(3-2\delta+2k\epsilon)}{2} < 0$.
Case 3: Some peer $\pi_z \in \Pi_z$ connects to Π^c_{j*}: unlike in the previous case, π_z prefers switching its link from Π^c_{j*} to a peer in Π^b_{j*} in the absence of a link from Π_y to Π^b_{j*}. By doing so, it can reduce its cost by $\Delta_z(\pi^c_{j*} \rightarrow \pi^b_{j*}) \leq \frac{k(3+2k\epsilon)}{2+\delta} - \frac{k(3+\delta)}{2} = k(-5\delta - \delta^2 + 4k\epsilon) < 0$.
Case 4: Some peer $\pi_z \in \Pi_z$ connects to Π^c_{j*} and some peer $\pi_y \in \Pi_y$ connects to Π^b_{j*}: in this configuration, peer π_y benefits from removing its link to Π^b_{j*}. The decrease of its costs is $\Delta_y(-\pi^b_{j*}) < -\alpha - k + \frac{k(3.1+2k\epsilon)}{2} < 0$.
Finally, since none of these four cases is a Nash equilibrium, the proof is concluded. □

9.5 Concluding Remarks

The analysis of our locality game reveals that the efficiency of p2p topologies can suffer if peers act selfishly. Moreover, our results indicate that topologies may degrade more severely when selfish peers value maintenance cost relatively higher than latency costs. Finally, it has been shown that it is generally a hard problem to decide whether a p2p system can stabilize if peers select their neighbors in a selfish manner.

So far, we did not conduct any measurement studies to verify whether such phenomena also exist in reality. Many theoretic questions are left for future research as well. For instance, it would be interesting to know whether incentive mechanisms can be designed such that the resulting topologies have desirable properties, e.g., are hypercubic or pancake networks. Some of our assumptions should also be weakened; e.g., a peer may not have complete knowledge of the other peers' states. Finally, we have not investigated how optimal topologies (with respect to our social cost function) can be computed, and approximate or mixed Nash equilibria have not been considered yet either.

Chapter 10

Impact of Malicious Players

The last chapter has investigated the effects of selfishness in unstructured peer-to-peer networks. However, a p2p system is also threatened by other forms of non-cooperative behavior. In the following, we will extend the game-theoretic framework to take malicious behavior into account. For instance, a malicious player may seek to harm the network, in order to prevent the unlawful distribution of a copyrighted movie.

To study the impact of malicious players on a given system formally, we introduce the *Price of Malice* of selfish systems. The Price of Malice is a ratio that expresses how much the presence of malicious players deteriorates the social welfare of a system consisting of selfish players. More technically, the Price of Malice is the ratio between the social welfare or performance achieved by a selfish system containing a number of *Byzantine*[1] players, and the social welfare achieved by an entirely selfish society.

It is interesting to compare the Price of Malice with the notion of the Price of Anarchy. As described in Chapter 9, the Price of Anarchy captures the degradation of a socially optimal performance of a system due to selfish behavior of its users or participants. That is, the Price of Anarchy relates the social welfare generated by players acting in an egoistic manner to an optimal solution obtained by perfectly collaborating participants. The Price of Malice's reference point, on the other hand, is not a socially optimal welfare, but the welfare achieved by an entirely selfish system.

The Price of Anarchy and the Price of Malice are therefore two orthogonal measures that describe inherent properties of distributed, socio-economic systems. Specifically, a system may have a small Price of Anarchy, but a large Price of Malice, and vice versa. The fact that a system has a large Price of Anarchy indicates that it is necessary to design mechanisms (such as taxes or payment schemes) that force players to collaborate more effec-

[1] In this chapter, the terms "malicious" and "Byzantine" will be used interchangeably. Observe that this is not always correct [182], and that "malicious" is generally more exact. However, in order to be consistent with our publications, henceforth, both terms will be used.

tively. However, it is more difficult to improve or repair systems having a large Price of Malice, since malicious or Byzantine players do not respond to any rules or (financial) incentives. Often, the only solution is to defend the system against malicious intruders, or at least to ensure that the number of malicious players in the system remains small.

By introducing a model that formally comprises the notions of *Byzantine Nash equilibria*, the *Byzantine Price of Anarchy*, and the *Price of Malice*, we are able to analyze what happens in selfish systems if one or more players' aim is to hinder the system from working or to bog down its performance as much as possible.

The Price of Malice crucially depends on the amount of information the selfish players have about the presence and behavior of the Byzantine players, and how they respond to this information. In other words, the utility function which eventually defines the selfish players' reaction depends on how they *subjectively* perceive and judge the threat of Byzantine players. Hence, the utility of selfish players is computed using the *perceived expected cost* rather than the unknown actual cost. For example, it can be shown that in case of risk-averse players, the presence of Byzantine players may actually *improve* the social welfare compared to a situation where there are no Byzantine players at all. That is, there are situations where selfish players tend to be more willing to collaborate if they face higher risks. To the best of our knowledge, this is the first framework which allows for an *analytical quantification* of this so-called *Fear Factor*. Potentially, this in turn gives raise to several research questions in many areas including distributed systems, economics, politics, or sociology. Besides studying Byzantine players who aim at minimizing the performance of a system (Price of Malice), we also raise the question of *stability*. Particularly, we are interested in the question, how many Byzantine players suffice in order to prevent the system from stabilizing.

In this chapter, we investigate a concrete example where selfish and Byzantine players interact. In this simple game, we consider a network of nodes, where each node (or peer) can choose between paying for inoculation, or risking to get infected by a virus. After the nodes have made their choices, a virus starts at some random node and propagates iteratively to all neighboring nodes which are not inoculated.

As a motivation for this game, consider the peer-to-peer Internet telephony tool Skype, and regard the different Skype instances as nodes in a graph. Two nodes are connected in this graph iff they are contained in each other's contact list. Each user can decide whether she likes to buy an anti-virus software or not. A p2p worm is then assumed to propagate along the contacts stored by the Skype instance, similarly to the *M-Worm:W32/Pykse.A*[2] virus .

[2] See http://news.softpedia.com/news/Skype-Attacked-By-Fast-Spreading-Virus-52039.shtml.

10.1 Framework

We present our model in two steps. First, we discuss the virus inoculation game derived from [24]. Subsequently, we introduce our framework of Byzantine game theory including the definition of the Price of Malice.

10.1.1 Virus Inoculation Game

Similarly to [24], we model the virus inoculation game as a scenario with n strategic players each of whom corresponds to a node in an undirected grid $G[r, c]$ of r rows and c columns.[3] Henceforth, we will refer to the upper left corner of the grid as $G[0, 0]$, i.e., indices start with 0.

Each node i has two choices: either do nothing and risk infection by a virus, or inoculate itself by installing anti-virus software. For a node, installing the anti-virus software has the obvious advantage that it becomes immune against infection. On the other hand, the process of installing the software entails a cost in terms of money and/or time. Hence, a strategic player may or may not opt for inoculation depending on which choice maximizes his own utility.

The nodes' choices can be summarized by a strategy profile $\overrightarrow{a} \in \{0, 1\}^n$, where $a_i = 1$ signifies that node i installs the anti-virus software, and $a_i = 0$ that it does not install it. We call nodes i with $a_i = 1$ *secure*, and denote the set of secure nodes by $I_{\overrightarrow{a}}$. After the nodes have made their choices, the adversary picks some node uniformly at random as a starting point for infection. Infection then propagates on the network graph and infects all non-secure nodes that are in the same non-secure connected component as the starting point of infection. Technically, we associate an *attack graph* $G_{\overrightarrow{a}} = G \setminus I_{\overrightarrow{a}}$ with $\overrightarrow{a}$. It is essentially the network graph in which all secure nodes and their incident edges are removed.

In this chapter, we consider the following costs: installing anti-virus software on a selfish node entails an inoculation cost of 1 at this node. If a selfish node does not inoculate and becomes infected, it suffers a loss equal to L. Therefore, the cost of a selfish node i can be summarized as follows:

$$cost_i(\overrightarrow{a}) = a_i + (1 - a_i) \cdot L \cdot \frac{k_i}{n},$$

where k_i/n is the probability that node i is infected, conditioned on the event that it does not install the anti-virus software. Thereby, k_i is the size of the connected component containing i in $G_{\overrightarrow{a}}$. Finally, the *social cost* of a strategy profile $\overrightarrow{a}$ is the sum of all individual costs, i.e., $Cost(\overrightarrow{a}) = \sum_{j \in \mathcal{S}} cost_j(\overrightarrow{a})$, where S denotes the set of all selfish players. When the strategy profile $\overrightarrow{a}$ is clear from the context, we sometimes use abbreviations $cost_i$ and $Cost$ to denote individual cost and social cost, respectively.

[3] Our results can be generalized to other highly regular, low-dimensional graphs such as the two-dimensional torus, i.e., a grid that wraps around at the boundaries.

10.1.2 Byzantine Game Theory

In order to understand the impact of malicious players on the selfish system, we extend the virus inoculation game with malicious players. Formally, there are n nodes in the network. Of these n nodes, b are malicious *Byzantine nodes* that do not strive for minimizing their own costs. Instead, the goal of these Byzantine nodes is to deteriorate the overall system performance as much as possible, i.e., to maximize the resulting social cost of the solution. The remaining $s := n-b$ nodes are *selfish* and aim at maximizing their own utility. We denote the set of Byzantine and selfish players as B and S, respectively. It holds that $b := |B|$, $s := |S|$, and $n = s + b$.

While selfish nodes behave as discussed in Chapter 10.1.1, we assume that the Byzantine nodes pursue the following strategy: they claim to be inoculated (i.e., they proclaim their strategy to be $a_i = 1$), but actually they are not. In order to emphasize that Byzantine nodes are only seemingly secure, we denote the set of really inoculated and secure selfish nodes by $I_{\overrightarrow{a}}^{self}$. The attack graph resulting from strategy profile $\overrightarrow{a}$ is then $G_{\overrightarrow{a}} = G - I_{\overrightarrow{a}}^{self}$. This is the network graph without secure, selfish nodes, but including all Byzantine nodes. We can therefore define the individual cost incurred at a selfish node $i \in S$ as follows.

Definition 10.1 (Actual Individual Cost)**.** *We define the (actual) individual cost $cost_i(\overrightarrow{a})$ of a node $i \in S$ as*

$$cost_i(\overrightarrow{a}) := a_i + (1 - a_i) \cdot L \cdot \frac{k_i}{n},$$

where k_i is the size of the connected component of node i in the attack graph $G_{\overrightarrow{a}}$.

Notice that in spite of its being equivalent to the corresponding definition in Chapter 10.1.1, we call this cost *actual individual cost.* This is to emphasize the fact that selfish players may not know about the existence of Byzantine players, and therefore, they are unable to compute their actual individual cost. Even if they are aware of the malicious players' existence, they might not know the Byzantine players' exact locations or strategies. In other words, with the addition of Byzantine players, selfish nodes no longer have a *perfect knowledge* about the network and its nodes' choices.

In case of imperfect information, a node might deal with its uncertainty in different ways. For example, a node might be risk averse and act in a conservative manner. These observations imply that before the location and strategies of Byzantine players are revealed (i.e., before the virus infection occurs), a selfish player i experiences a *perceived individual cost* $\widehat{cost_i}(\overrightarrow{a})$. This perceived cost can differ from the *actual individual cost* $cost_i(\overrightarrow{a})$ a node eventually has to pay.

Definition 10.2 (Perceived Individual Cost)**.** *Consider a selfish game with Byzantine players in which selfish players have imperfect knowledge about the existence, location, or the strategy of Byzantine players. In this case,* *the perceived individual cost $\widehat{cost}_i(\overrightarrow{a})$ of a selfish player i captures the cost expected by player i given his knowledge about the Byzantine players. This cost depends on the underlying model.*

The strategic decisions of selfish players can only be based on the *perceived cost* (not on their actual individual costs), as the actual individual cost can only be computed once the locations and strategies of Byzantine players are revealed. In this chapter, we will study the following two basic models.

Definition 10.3 (Oblivious)**.** *In the* oblivious model*, selfish players are not aware of the existence of Byzantine players. That is, selfish players assume that all other players in the system are selfish as well.*

Definition 10.4 (Non-Oblivious)**.** *In the* non-oblivious model*, selfish players know about the existence of Byzantine players. Specifically, we assume that every selfish player knows b, the number of Byzantine players in the system, but he does not know about these players' exact locations or strategies. Moreover, we assume that selfish players are highly risk averse in the sense that they aim at minimizing their maximal individual cost. Let $\mathcal{D}$ be the set of possible distributions of Byzantine players among all players. A selfish player i experiences a perceived individual cost of*

$$\widehat{cost}_i(\overrightarrow{a}) := \max_{d \in \mathcal{D}} \{cost_i(\overrightarrow{a}, d)\},$$

where $cost_i(\overrightarrow{a}, d)$ denotes the actual costs of i if the Byzantine players are distributed according to $d \in \mathcal{D}$.

In the virus inoculation game, and in an oblivious model, the perceived cost is typically smaller than the actual cost: a node $i \in S$ does not take into consideration the Byzantine nodes which may increase the size of i's attack component. In the non-oblivious risk-averse model on the other hand, a node actually overestimates its expected actual cost by considering a worst-case scenario: a selfish player assumes that the Byzantine nodes are—from his individual point of view—distributed in a worst-case fashion among all players. Therefore, the perceived individual cost may be larger than the actual cost.

Since our goal is to understand the impact of malicious behavior on a system of selfish players, the cost of Byzantine players is not included in the social cost. If it was, it would in general be easy for Byzantine players to arbitrarily deteriorate the social welfare of a system by simply increasing their own costs as much as possible. Moreover, as Byzantine players are malicious anyway, there is no particular reason why the overall system should care about these players' costs.

The total *social cost* $Cost(\overrightarrow{a})$ of a strategy is defined as the sum of the (actual) individual costs of all selfish players. Since each node in the same connected component of $G_{\overrightarrow{a}}$ has the same probability of infection, the l_i selfish nodes in the i^{th} attack component face a loss of $l_i \cdot (Lk_i/n)$ if the component is infected.

Definition 10.5 (Social Cost). *The social cost is given by the sum of the actual individual costs of selfish players*

$$Cost(\overrightarrow{a}) = \sum_{j \in \mathcal{S}} cost_j(\overrightarrow{a}) = \underbrace{|I_{\overrightarrow{a}}^{self}|}_{inoculation\ cost} + \underbrace{\frac{L}{n}\sum_{i=1}^{l} k_i l_i}_{infection\ cost},$$

where $k_1, k_2, \ldots, k_l$ are the sizes of the components in $G_{\overrightarrow{a}}$, and $l_1, l_2, \ldots, l_l$ are the sizes of the same components without counting the Byzantine nodes. We refer to the cost due to inoculation as the inoculation cost $Cost_{inoc}$*, and to the cost due to the virus infections as the* infection cost $Cost_{inf}$.

The social cost of a setting where all nodes perfectly collaborate, i.e., where there are neither selfish nor Byzantine nodes, is called the *social optimum.*

Definition 10.6 (Optimal Social Cost). *The optimal social cost $Cost_{OPT}$ is the sum of all the players' actual individual costs in case of perfect collaboration.*

Recall that the Nash equilibrium describes a situation where no selfish node has an incentive to unilaterally change its strategy. In the following, we extend the definition of a Nash equilibrium to incorporate Byzantine nodes. The *Byzantine Nash equilibrium* (BNE) describes a configuration where no selfish player can reduce his *perceived* cost by changing his strategy, given that the strategies of all other players are fixed.[4]

Definition 10.7 (Byzantine Nash Equilibrium (BNE)). *Let $\overrightarrow{a}[i|x]$ be the strategy vector that is identical to $\overrightarrow{a}$ except for the i^{th} component a_i which is replaced by x. In a Byzantine Nash equilibrium, no selfish player $i \in \mathcal{S}$ has an incentive to change his strategy if the strategies of all other (selfish and Byzantine) players are fixed, i.e.,*

$$\forall i \in \mathcal{S} : \widehat{cost}_i(\overrightarrow{a}) \leq \widehat{cost}_i(\overrightarrow{a}[i|a_i']),$$

for every possible strategy a_i'.

While the Byzantine Nash equilibrium must be defined by the *perceived* individual costs, the resulting social cost is determined by the *actual* costs. After all, it is the actual individual costs that players will eventually have to pay. In the following, we will refer to the social cost of the *worst Byzantine Nash Equilibrium* of a problem instance I as $Cost_{BNE}(I, b)$.

It is well-known that selfish and Byzantine players often interact in a manner that yields suboptimal solutions. The degree of degradation resulting from selfish and Byzantine players compared to the social optimum is captured by the *Price of Byzantine Anarchy.*

[4]Notice that we do not define the Byzantine Nash equilibrium with *actual* individual costs, because they are not known to the players.

Definition 10.8 (Price of Byzantine Anarchy). *The Price of Byzantine Anarchy captures how much worse a Byzantine Nash equilibrium can be compared to a collaborative optimal solution. More formally, in a scenario with b Byzantine players, the Price of Byzantine Anarchy $PoB(b)$ is the ratio between the worst-case social cost of a Byzantine Nash equilibrium divided by the minimal social cost, i.e., for all problem instances I,*

$$PoB(I, b) = \frac{\max_{BNE} Cost_{BNE}(I, b)}{Cost_{OPT}(I)}.$$

Note that in the absence of Byzantine players—i.e., if the system consists of selfish players only—the Price of Byzantine Anarchy is equivalent to the well-known Price of Anarchy (PoA) studied in classic game theory. Specifically, it holds that $PoA = PoB(0)$.

With these definitions, we are ready to define the *Price of Malice* which describes the degree of sub-optimality resulting from malicious Byzantine players in an otherwise selfish system. A high Price of Malice indicates that an economic system is particularly vulnerable to malicious attacks. On the other hand, if the Price of Malice is low, the system consisting of selfish players is stable enough to tolerate malicious participants. Clearly, the degree of degradation may depend on the number of Byzantine players in the game. Hence, the Price of Malice is a function of b.

Definition 10.9 (Price of Malice). *The Price of Malice captures the ratio between the worst Byzantine Nash Equilibrium with b malicious players and the Price of Anarchy in a purely selfish system. Formally, for problem instance I,*

$$PoM(I, b) = \frac{PoB(I, b)}{PoB(I, 0)}.$$

As will be discussed in Chapter 10.2.4, we may also speak of the inverse of the Price of Malice as the game's *Fear Factor* $\Psi(b)$. That is, a game's Fear Factor is given by $\Psi(b) := 1/PoM(b)$.

10.2 Virus Game Analysis

In order to derive results for the Price of Malice in various models, we have to establish structural properties of Nash equilibria and the social optimum in the virus inoculation game. We begin with a simple characterization of Nash equilibria if there are no Byzantine nodes. The following lemma is derived from the analogous lemma in [24].

Lemma 10.1. *In a pure Nash equilibrium $\vec{a}$, it holds that (a) every component in the attack graph $G_{\vec{a}}$ has a size of at most n/L, and (b) inserting any secure node into $G_{\vec{a}}$ yields a component size of at least n/L.*

Lemma 10.1 implies that if $L \geq n$, all nodes will inoculate in the Nash equilibrium. Therefore, for the rest of this chapter, we assume that $L < n$.

10.2.1 Social Optimum

If the inoculation strategies of the individual nodes are planned by a benevolent centralized coordinator, the welfare of the system is maximized. In the following, we will derive an asymptotically tight bound on the cost of this social optimum. Throughout this section, perceived costs equal actual costs because when studying the social optimum, we do not consider Byzantine players, i.e., $b = 0$ and therefore $s = n$.

Theorem 10.2. *The optimal social cost if all players in S act altruistically is $Cost_{OPT} \in \Theta(s^{2/3}L^{1/3})$. More specifically,*

$$\frac{1}{3}\sqrt{\pi} \cdot s^{2/3}L^{1/3} \ \le\ Cost_{OPT} \ \le\ 4s^{2/3}L^{1/3}.$$

Proof. We prove the upper and lower bound in turn.

Lower Bound: If all nodes collaborate to achieve the optimal solution, it holds that $l_i = k_i$ and hence, the social cost is given by

$$Cost \ =\ |I_{\vec{a}}| + \frac{L}{n}\sum_{i=1}^{l} k_i^2,$$

where $|I_{\vec{a}}|$ is the number of inoculated nodes, and the k_i's are the sizes of the components in the attack graph. This sum is minimized when all k_i are of equal size, say size K. While each secure node has a cost of 1, every other node has an expected cost of $L \cdot K/n$. Hence, setting $\gamma := |I_{\vec{a}}|$ and because $s = n$, the optimal social cost can be bounded as

$$Cost_{OPT} \ \ge\ \gamma + (s-\gamma)\left(\frac{LK}{s}\right). \tag{10.1}$$

A relationship between γ and K follows from a simple geometric argument: if a component in the attack graph is of size K, the number of inoculated nodes at the component's border must be at least $2\pi\sqrt{\frac{K}{\pi}} = 2\sqrt{\pi K}$ (circumference of a disk with volume K). As the total number of such components is at least $\frac{s-\gamma}{K}$ and as each inoculated node can be on the border of at most two components, γ can be expressed as

$$\gamma \ \ge\ \frac{s-\gamma}{K} \cdot 2\sqrt{\pi K} \cdot \frac{1}{2} \ =\ (s-\gamma)\sqrt{\frac{\pi}{K}}.$$

By solving this inequality for γ, it follows that $\gamma \ \ge\ s \cdot \sqrt{\pi/K}/(1+\sqrt{\pi/K})$. On the other hand, it can be observed that in the optimal solution, for $s > L$, no node is inoculated if all its four neighbors are inoculated. From this, it can be derived that in an optimal solution, $\gamma \le \frac{s}{2}$. Plugging these two bounds into Inequality (10.1), the optimal social cost is at least

$$Cost_{OPT} \ \ge\ s \cdot \frac{\sqrt{\pi/K}}{1+\sqrt{\pi/K}} + \frac{LK}{2}.$$

The first term of the above expression is monotonously decreasing in K in the range $0, \dots, s$, whereas the second one is monotonously increasing. Therefore, taking the minimum of the two terms for a specific K yields a lower bound on $Cost_{OPT}$. When setting

$$K := \frac{2}{3}\sqrt{\pi} \cdot \left(\frac{s}{L}\right)^{2/3},$$

the second term yields $\frac{1}{3}\sqrt{\pi} \cdot s^{2/3}L^{1/3}$. The first term evaluates to $\sqrt{3/2} \cdot \sqrt[4]{\pi}/(1+\sqrt{3/2}\cdot\sqrt[4]{\pi})s^{2/3}L^{1/3} > \frac{1}{3}\sqrt{\pi}\cdot s^{2/3}L^{1/3}$. Consequently, we obtain the following lower bound on the cost of the social optimum:

$$Cost_{OPT} \quad \geq \quad \frac{1}{3}\sqrt{\pi} \cdot s^{2/3}L^{1/3} \in \Omega(s^{2/3}L^{1/3}).$$

Upper Bound: Having established a lower bound on the optimal social cost, we now explicitly construct a solution that is asymptotically optimal and proves the tightness of the above lower bound. Given an arbitrary grid $G[r,c]$, we inoculate the nodes as follows. Let $K := (s/L)^{2/3}$. We secure all nodes in the columns $G[\cdot, i\sqrt{K}]$ for $i \in \{1, ..., \lfloor c/(\sqrt{K}+1)\rfloor\}$ and rows $G[i\sqrt{K}, \cdot]$ for $i \in \{1, ..., \lfloor r/(\sqrt{K}+1)\rfloor\}$. Consequently, all attack components are of size at most $\sqrt{K} \times \sqrt{K} = K$ as illustrated in Figure 10.1 (*Left*). Hence, the total infection cost is at most $L \cdot (s - |I_{\vec{a}}|)\frac{K}{s} < LK = s^{2/3}L^{1/3}$.

It remains to bound the inoculation cost. In an ideal setting where the components perfectly fit into $G[r,c]$ without leftovers, it holds that for each component of size K in the attack graph, there are exactly $2\sqrt{K}+1$ inoculated nodes. Let X denote the number of components. It holds that $X \cdot (K + 2\sqrt{K}+1) = s$ and therefore, when plugging in the definition of K, $X = s/[(s/L)^{2/3} + 2(s/L)^{1/3} + 1]$. The number of inoculated nodes γ is at most

$$\begin{aligned} \gamma &\leq X \cdot (2\sqrt{K}+1) \leq \frac{s(2\sqrt{K}+1)}{\left(\frac{s}{L}\right)^{2/3} + 2\left(\frac{s}{L}\right)^{1/3} + 1} \\ &< s^{1/3}L^{2/3} \cdot \left(2\left(\frac{s}{L}\right)^{1/3} + 1\right) = 2s^{2/3}L^{1/3} + s^{1/3}L^{2/3} \\ &\leq 3s^{2/3}L^{1/3}. \end{aligned}$$

Combining the infection and inoculation costs, we can bound the optimal social cost by

$$Cost_{OPT} < s^{2/3}L^{1/3} + 3s^{2/3}L^{1/3} = 4s^{2/3}L^{1/3}.$$

□

10.2.2 Price of Anarchy

The Price of Anarchy compares the social cost of the worst Nash equilibrium (without Byzantine nodes) to the minimal social cost. In the upcoming

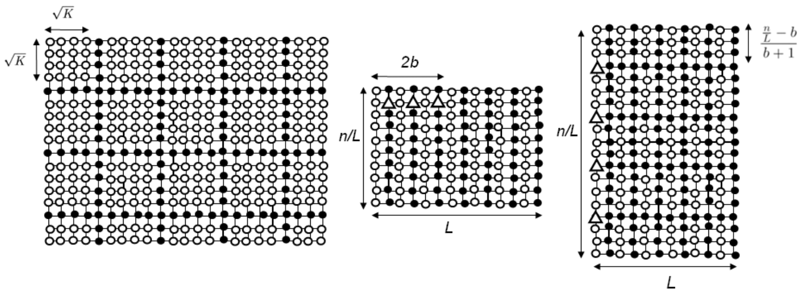


Figure 10.1: *Left:* Upper bound for social optimum. White nodes are insecure, black nodes are secure. *Middle:* Byzantine Nash equilibrium for $G[n/L, L]$ for the oblivious model. Insecure Byzantine nodes are denoted by white triangles. They are located in a way that may yield an attack component of size $(b+1)n/L+b$. *Right:* Example with large social cost for the non-oblivious, risk-averse model.

section, we will first compute $Cost_{NE}$, which is the maximal cost of any Nash equilibrium. Together with the bound for the social optimum in Chapter 10.2.1, the Price of Anarchy will follow.

Lemma 10.3. *The social cost of the worst Nash equilibrium is $Cost_{NE} = \Theta(s)$.*

Proof. First, we show that $Cost_{NE} = \Omega(s)$. Consider a grid $G[s/L, L]$ consisting of an even number of L rows of size s/L. Assume that columns $G[\cdot, 2i]$ for $i \in \{0, 1, ..., L/2-1\}$ consist of insecure nodes only, while all nodes in the remaining rows are secure. Since all attack components have size s/L, according to Lemma 10.1, this situation constitutes a Nash equilibrium. Observe that every second row is inoculated, engendering an inoculation cost of $s/2$. Moreover, with probability $1/2$, the virus starts at an insecure node, yielding infection cost $s/L \cdot L$. The social cost is therefore $Cost_{NE} = s/2 + 1/2 \cdot s/L \cdot L = s$.

It remains to show that $O(s)$ is an upper bound for any Nash equilibrium. Since at most each of the $s = n$ nodes can be inoculated, the inoculation cost cannot exceed s. By Lemma 10.1, we also know that the infected component's size is at most s/L, entailing a total infection cost of at most s as well. Hence, $Cost_{NE} \leq 2s$, and the claim holds. $\square$

By Theorem 10.2 and Lemma 10.3, we get the following result.

Theorem 10.4. *For the Price of Anarchy (PoA), it holds that*

$$\frac{1}{4} \cdot \left(\frac{s}{L}\right)^{1/3} \leq PoA \leq \frac{6}{\sqrt{\pi}} \cdot \left(\frac{s}{L}\right)^{1/3}.$$

Proof. As for the upper bound, it holds that

$$PoA = \frac{Cost_{NE}}{Cost_{OPT}} \leq \frac{2s}{\frac{1}{3}\sqrt{\pi} \cdot s^{2/3}L^{1/3}} \leq \frac{6s^{1/3}}{\sqrt{\pi} \cdot L^{1/3}}$$

and as for the lower bound, we have $PoA \geq \frac{s}{4 \cdot s^{2/3}L^{1/3}}$. □

10.2.3 Oblivious Model

We begin our study of the Price of Malice with the oblivious model in which players are clueless about the existence of malicious players in the system (cf Chapter 11.1). As a consequence, it follows that—since nodes underestimate the attack components' sizes—the nodes' perceived individual costs are smaller than the actual individual costs. It turns out that in the presence of Byzantine nodes, the social costs increase in the number of Byzantine nodes.

Lemma 10.5. *In the oblivious model, the social cost is at least* $Cost_{BNE} \in \Omega(s + \frac{nb^2}{L})$ *for* $b < \frac{L}{2} - 1$*, and* $Cost_{BNE} \in \Omega(sL)$ *otherwise.*

Proof. Consider again a grid $G[n/L, L]$ with n/L rows and L columns, where every second column consists of secure nodes only. For simplicity, let L be even. Suppose that in the first b secure columns there is one Byzantine node each, see Figure 10.1 (*Middle*). In case $b \geq \frac{L}{2} - 1$, every secure column that separates two insecure columns contains one Byzantine node. The remaining Byzantine nodes can be placed at arbitrary places in the secure columns. Because selfish nodes are not aware of the existence of Byzantine nodes in the network, the perceived cost is $\widehat{cost}_i = 1$ for inoculated nodes, and $\widehat{cost}_i = \frac{n/L}{n} \cdot L = 1$ for the other selfish nodes. Hence, the situation constitutes a *Byzantine Nash equilibrium.*

For computing the social costs of this Byzantine Nash equilibrium, we distinguish two cases, depending on whether the number of Byzantine nodes is smaller than $L/2 - 1$ or not. For the first case, assume that $b \geq L/2 - 1$. Because there is at least one Byzantine node in every secure column that separates two insecure columns has least one Byzantine node, all selfish and Byzantine players form one large attack component. Consequently, each insecure selfish node $i \in S$ is infected with probability 1 and therefore $Cost_{BNE} \geq s \cdot L$.

For the second case, assume that $b < L/2 - 1$. Each of the first secure columns contains exactly one Byzantine node. Since L is even, there are $s/2 - b$ secure nodes, and hence the inoculation cost is $s/2 - b$. With probability $((b+1)n/L + b)/n$, the infection starts at an insecure or a Byzantine node of an attack component of size $(b+1) \cdot n/L$, yielding a cost of $(b+1) \cdot n/L \cdot L = n(b+1)$. Moreover, with probability $(s/2 - (b+1)n/L)/n$, an insecure column of size n/L is hit. Thus, for $b < L/2 - 1$, we get the following lower bound on the social cost:

$$\begin{aligned}
Cost_{BNE} &= \left(\frac{s}{2}-b\right)+\frac{\frac{(b+1)n}{L}+b}{n}\cdot n(b+1)+\\
&\quad +\frac{\frac{s}{2}-(b+1)\frac{n}{L}}{n}\cdot\frac{n}{L}\cdot L\\
&= s+\frac{nb^2}{L}+\frac{nb}{L}+b^2\ \in\ \Omega\left(s+\frac{nb^2}{L}\right).
\end{aligned}$$

□

Lemma 10.6. *In the oblivious model, the social cost is at most* $Cost_{BNE} \in O\left(\min\{sL, s+\frac{b^2n}{L}\}\right)$.

Proof. Since at most every selfish node can be inoculated, it is clear that $Cost_{inoc} = O(s)$. It remains to study the infection cost. The infection cost of a node in some component i is L times the probability of this component being hit by the virus, i.e., $L \cdot k_i/n$. Hence, the total infection cost is given by

$$Cost_{inf} = \sum_i l_i \cdot \frac{k_i}{n}\cdot L = \frac{L}{n}\sum_i l_i \cdot k_i,$$

where k_i is the size of the attack components (including Byzantine nodes), and l_i is the number of selfish nodes in this component. In order to upper bound $Cost_{inf}$, let S_{Byz} denote the set of components in the attack graph which contain at least one Byzantine node, and let $S_{\overline{Byz}}$ be the remaining components. We can rewrite the equation above as

$$Cost_{inf} = \frac{L}{n}\cdot\left[\sum_{i\in S_{Byz}} l_i\cdot k_i + \sum_{i\in S_{\overline{Byz}}} l_i\cdot k_i\right],$$

that is, we consider the infection cost of components with at least one Byzantine node separately from the remaining "malicious player-free" components. In the following, let

$$Cost_{inf}^{Byz} := \frac{L}{n}\sum_{i\in S_{Byz}} l_i k_i \qquad Cost_{inf}^{\overline{Byz}} := \frac{L}{n}\sum_{i\in S_{\overline{Byz}}} l_i k_i.$$

We have to prove that neither $Cost_{inf}^{Byz}$ nor $Cost_{inf}^{\overline{Byz}}$ exceeds $O(s+\frac{b^2n}{L})$.

As we have shown in the proof of Lemma 10.3 in Chapter 10.2.2, the total infection cost of a network consisting only of selfish nodes cannot exceed s. Because in our case nodes are oblivious about the existence of Byzantine nodes, attack components without Byzantine nodes behave like in an entirely selfish environment. Therefore, $Cost_{inf}^{\overline{Byz}} \in O(s)$.

It remains to compute the infection cost of those attack components which include at least one Byzantine node. Let b_i be the number of Byzantine nodes in the i^{th} component in S_{Byz}, and note that $\sum_i b_i = b$. By Lemma 10.1, we know that in the absence of Byzantine nodes, the size of an attack component is at most $k_i \le n/L$. Therefore, one Byzantine node can increase a component by at most n/L nodes plus itself. From this it follows that the size of an attack component i is bounded by

$$k_i \;\le\; (b_i+1)\cdot\frac{n}{L}+b_i, \quad \text{and} \quad l_i \;\le\; (b_i+1)\cdot\frac{n}{L}.$$

Using this relationship between b_i and the size of the attack component, we can bound $Cost_{inf}^{Byz}$ as

$$\begin{aligned}
Cost_{inf}^{Byz} &= \frac{L}{n}\sum_{i\in S_{Byz}} l_i\cdot k_i \\
&\le \frac{L}{n}\sum_{i\in S_{Byz}} \left[(b_i+1)\cdot\frac{n}{L}\cdot\left((b_i+1)\cdot\frac{n}{L}+b_i\right)\right] \\
&= \sum_{i\in S_{Byz}} \left[(b_i+1)^2\frac{n}{L}+b_i(b_i+1)\right] \\
&< \sum_{i\in S_{Byz}} \left[(b_i+1)^2\left(\frac{n}{L}+1\right)\right] \\
&= \left(\frac{n}{L}+1\right)\cdot\sum_{i\in S_{Byz}}(b_i+1)^2.
\end{aligned}$$

Given the constraint that $b_i \ge 1$ for every b_i, and because $\sum_i b_i = b$, the above convex function assumes its maximum for a single positive $b_i = b$. Consequently,

$$\begin{aligned}
Cost_{inf}^{Byz} &\le \left(\frac{n}{L}+1\right)\cdot\sum_{i\in S_{Byz}}(b_i+1)^2 \\
&\le \left(\frac{n}{L}+1\right)\cdot(b+1)^2 \;\in\; O\left(\frac{b^2 n}{L}\right).
\end{aligned}$$

On the other hand, it clearly holds that at most every selfish node can be infected and hence, $Cost_{inf}^{\overline{Byz}} + Cost_{inf}^{Byz} \le sL$. The proof is concluded by adding the upper bounds for $Cost_{inoc}$, $Cost_{inf}^{\overline{Byz}}$, and $Cost_{inf}^{Byz}$. □

Combining Lemmas 10.5 and 10.6 leads to the following theorem that captures the social cost in the virus inoculation game in the presence of b Byzantine players among selfish, oblivious nodes.

Theorem 10.7. *The social cost in a Byzantine Nash equilibrium with b Byzantine nodes in the oblivious model is $Cost_{BNE} \in \Theta(s+\frac{b^2 n}{L})$, for $b < \frac{L}{2}-1$, and $Cost_{BNE} \in \Theta(sL)$, otherwise.*

Proof. In both cases, the lower bound follows from Lemma 10.5. As for the upper bound, note that for $b < L/2 - 1$ and due to $L \leq n = s + b$, it holds that $b < (s+b)/2$ and therefore, $b < s$. Then, the term $s + b^2 n/L$ asymptotically cannot exceed the term sL and therefore, the claim follows. As for the second case, note that for $b \geq \frac{L}{2} - 1$, the term sL is asymptotically smaller or equal to $s + b^2 n/L$. □

Finally, we can derive tight bounds on the Price of Byzantine Anarchy and the Price of Malice by bringing together the results of Theorems 10.2, 10.4, and 10.7.

Theorem 10.8. *In the virus inoculation game with b Byzantine nodes among selfish, oblivious nodes, the Price of Byzantine Anarchy and the Price of Malice are*

$$\begin{aligned} PoB(b) &\in \Theta\left(\left(\frac{s}{L}\right)^{1/3}\left(1 + \frac{b^2}{L} + \frac{b^3}{sL}\right)\right) \quad \textit{and} \\ PoM(b) &\in \Theta\left(1 + \frac{b^2}{L} + \frac{b^3}{sL}\right) \end{aligned}$$

for $b < \frac{L}{2} - 1$. Otherwise, it holds that

$$PoB(b) \in \Theta\left(s^{1/3}L^{2/3}\right) \quad \textit{and} \quad PoM(b) \in \Theta\left(L\right).$$

Proof. Consider the case $b < \frac{L}{2} - 1$. For the Price of Byzantine Anarchy, we have $PoB(b) = \frac{Cost_{BNE}}{Cost_{OPT}} = \frac{\Theta(s + \frac{b^2(b+s)}{L})}{\Theta(s^{2/3}L^{1/3})} \in \Theta\left(\left(\frac{s}{L}\right)^{1/3} \cdot \left(1 + \frac{b^2}{L} + \frac{b^3}{sL}\right)\right)$. From this, the Price of Malice is computed as follows: $PoM(b) = \frac{PoB(b)}{PoA} \in \Theta\left(1 + \frac{b^2}{L} + \frac{b^3}{sL}\right)$. The case $b \geq \frac{L}{2} - 1$ follows along the same lines by plugging in the corresponding expressions of Theorem 10.7. □

Our results on the Price of Malice in the oblivious case support the intuition that in the absence of knowledge about the existence of Byzantine players, the quality of the global solution (i.e., the resulting social cost) deteriorates as the number of malicious players increases. In the next section, we will show that the situation may change as soon as selfish players are *aware* of the existence of Byzantine players.

10.2.4 Non-oblivious Model

Having studied the oblivious model, we now turn our attention to the non-oblivious case in which selfish players are aware of the existence of Byzantine players. If selfish nodes knew about the exact locations of Byzantine nodes, they would be able to compute their optimal choice exactly. If selfish nodes only know the *number* of Byzantine nodes in the system, however, the optimal strategy of a player becomes more complex, and the impact on the social cost more interesting. Specifically, it turns out that in this non-oblivious case, the "Fear Factor" may actually encourage players to act less selfishly and cooperate. Put differently, there may be settings in which the existence of Byzantine players helps to improve the global social cost, rendering the Price of Malice less than 1.

Observe that in the non-oblivious case, every selfish node inoculates if $b \geq \frac{n}{L}$, implying a social cost of s. If $b < \frac{n}{L}$, the resulting social costs are bounded by the following lemma.

Lemma 10.9. *For $b < \frac{n}{2L}$, the social cost in a Byzantine Nash equilibrium in case of non-oblivious, risk-averse players with b Byzantine nodes is at least*

$$Cost_{BNE} \geq \frac{s}{2} + \frac{bL}{4}.$$

For all values of b, it holds that $Cost_{BNE} \geq \frac{s}{2}$.

Proof. We start with the more interesting case $b < \frac{n}{2L}$. Consider a grid with L columns each containing n/L nodes. All nodes in columns $2i + 1$ for $i = 0, 1, \ldots, \frac{L}{2} - 1$ and all nodes in rows $j \cdot \frac{n/L-b}{b+1}$ for $j = 1, 2, \ldots, b$ are inoculated. That is, as illustrated in Figure 10.1 (*Right*), each component of insecure selfish nodes is of size $\frac{n/L-b}{b+1}$.

First, we show that this configuration constitutes a Byzantine Nash equilibrium in the risk-averse, non-oblivious case with b Byzantine nodes. Consider an *insecure node* in some column i. If all b secure nodes in this column are Byzantine, the size of the resulting attack component is $(n/L - b)/(b + 1) \cdot (b + 1) + b = n/L$. Hence, i's perceived infection cost is

$$\widehat{cost}_i = L \cdot \frac{(n/L - b)/(b + 1) \cdot (b + 1) + b}{n} = 1,$$

which equals the cost of inoculation. Next, consider an *inoculated selfish node* i and distinguish two cases. In the first case, i separates two components consisting of insecure selfish players and a change of i's strategy would merge two components of size $(n/L-b)/(b+1)$ into a single connected component of insecure selfish nodes. Every Byzantine node can connect another component of size $(n/L-b)/(b+1)$ (and itself) to the component containing i. Therefore, the size of the resulting attack component can be as large as

$$\left(2 \cdot \frac{\frac{n}{L} - b}{b+1} + 1\right) + \left(b \cdot \frac{\frac{n}{L} - b}{b+1} + b\right) = \frac{b+2}{b+1}\left(\frac{n}{L} - b\right) + b + 1 > \frac{n}{L} + \frac{1}{b+1}.$$

The perceived cost of i without inoculation is therefore

$$\widehat{cost}_i > L \cdot \frac{\frac{n}{L} + \frac{1}{b+1}}{n} = 1 + \frac{L}{n(b+1)} > 1.$$

In the second case, we consider a "crossing" node i that is located in the crossing of a secure row and column. Consider the column to the right (or to the left) of i. If all inoculated nodes in this column are Byzantine, the entire column plus node i becomes one large attack component. Hence, the perceived cost of i is

$$\widehat{cost}_i \; > \; L \cdot \frac{\frac{n}{L}+1}{n} \; > \; 1.$$

In other words, no selfish node has an incentive to change its strategy and the situation in Figure 10.1 (right) constitutes a Byzantine Nash equilibrium. In the sequel, we lower bound the social cost of this equilibrium under the assumption that all b Byzantine nodes are in column 1. Note that our construction guarantees that this is always possible if $b < \frac{n}{2L}$.

We start with the sum of the infection costs $Cost^0_{inf}$ of insecure nodes in column 0. The number of insecure, selfish nodes in this component is $\frac{n}{L} - b$. Hence, the expected sum of infection costs is

$$Cost^0_{inf} = \left(\frac{n}{L} - b\right) \cdot \frac{\frac{n}{L} - b + b}{n} \cdot L \; = \; \frac{n}{L} - b.$$

Let μ be the number of insecure nodes in columns $3, 5$, etc. The sum of the infection costs $Cost^r_{inf}$ of the remaining attack components (each being of size $(n/L - b)/(b+1)$) is

$$Cost^r_{inf} \; = \; \mu \cdot \frac{\frac{n}{L} - b}{n(b+1)} \cdot L \; > \; \mu \cdot \left(\frac{1}{b+1} - \frac{L}{n}\right).$$

Because the number of insecure nodes in these small attack components is $\mu = \frac{L-1}{2} \cdot \left(\frac{n}{L} - b\right)$, it follows that

$$\begin{aligned} Cost^r_{inf} \; &> \; \frac{L-1}{2} \cdot \left(\frac{n}{L} - b\right) \cdot \left(\frac{1}{b+1} - \frac{L}{n}\right) \\ &> \; \frac{1}{2(b+1)} \left(n - \frac{n}{L} - bL + b\right) - \frac{L}{2}. \end{aligned}$$

Finally, we also need to calculate the total inoculation cost of this topology. Clearly, all $s/2$ nodes in even columns are secure. (Recall that column and row indices start with 0.) Furthermore, b nodes in each odd column (except for the first column) are also inoculated. Hence, the total inoculation $Cost_{inoc}$ cost becomes

$$Cost_{inoc} \; = \; \frac{s}{2} + \frac{bL}{2} - b \; = \; \frac{s}{2} + b\left(\frac{L}{2} - 1\right).$$

Combining the various costs, the social cost of the Byzantine Nash equilibrium is

$$\begin{aligned} Cost_{BNE}(b) \; &\geq \; \frac{s}{2} + b\left(\frac{L}{2} - 1\right) + \frac{n}{L} - b \\ &\quad + \frac{1}{2(b+1)} \left(n - \frac{n}{L} - bL + b\right) - \frac{L}{2} \\ &\geq \; \frac{s}{2} + \frac{bL}{4} \end{aligned}$$

for $b \leq \frac{n}{2L}$ and $b \geq 3$.

Finally, note that if $b \geq \frac{n}{2L}$, at least half of the selfish nodes inoculate and hence, $Cost_{BNE}(b) \geq s/2$. □

With this lower bound on the social cost of a Byzantine Nash equilibrium, we can now derive the Price of Byzantine Anarchy as well as the Price of Malice for the non-oblivious, risk-averse model.

Theorem 10.10. *In the non-oblivious, risk-averse model with b Byzantine nodes, the Price of Byzantine Anarchy is at least*

$$PoB(b) \geq \frac{1}{8}\left(\left(\frac{s}{L}\right)^{1/3} + \frac{b}{2}\left(\frac{L}{s}\right)^{2/3}\right)$$

for $b < \frac{n}{2L}$. *For all b, it holds that* $PoB(b) \geq \frac{1}{8}(\frac{s}{L})^{1/3}$.

Proof. Lemma 10.9 gives us a lower bound on the social cost of a Byzantine Nash equilibrium in the non-oblivious, risk-averse model with b malicious nodes. On the other hand, we have seen in Lemma 10.2, that the optimal social cost is at most $4s^{2/3}L^{1/3}$. Hence,

$$PoB(b) \quad \geq \quad \frac{\frac{s}{2} + \frac{bL}{4}}{4s^{2/3}L^{1/3}} \;=\; \frac{1}{8}\left(\frac{s^{1/3}}{L^{1/3}} + \frac{bL^{2/3}}{2s^{2/3}}\right).$$

The second lower bound follows analogously. □

Theorem 10.11. *In the non-oblivious, risk-averse model with b Byzantine nodes, the Price of Malice is*

$$PoM(b) \geq \frac{\sqrt{\pi}}{48}\left(1 + \frac{bL}{2s}\right)$$

for $b < \frac{n}{2L}$. *For all b, it holds that* $PoM(b) \geq \frac{\sqrt{\pi}}{48}$.

Proof. In order to derive the Price of Malice, we can apply our bound from Theorem 10.10 and the upper bound on the Price of Anarchy established in Theorem 10.4. Specifically,

$$PoM(b) \;=\; \frac{PoB(b)}{PoA} \;\geq\; \frac{\frac{1}{8}\left(\left(\frac{s}{L}\right)^{1/3} + \frac{b}{2}\left(\frac{L}{s}\right)^{2/3}\right)}{\frac{6s^{1/3}}{\sqrt{\pi}\cdot L^{1/3}}}.$$

The theorem then follows from arithmetic simplifications. Again, the second lower bound follows in an analogous way. □

Discussion: From a technical point of view, this result shows that the Price of Malice can potentially be less than 1 in the non-oblivious model of the virus inoculation game. Intuitively, it is clear that in the presence of Byzantine players, nodes may be more willing to pay for inoculation. However, we find it interesting that the selfish players' awareness of the existence of malicious Byzantine players may lead to an *improvement* of the overall system behavior, i.e., the *social welfare*. Specifically, the existence (or even the threat!) of malicious Byzantine players can render it worthwhile for nodes to cooperate better.

This highlights the possible existence of a *Fear Factor*, which describes the gain of the overall social efficiency in a selfish system if selfish players are afraid of malicious, Byzantine individuals among them. This Fear Factor is determined by the ratio between the social cost of the worst Byzantine Nash equilibrium with b malicious players and the worst Nash equilibrium in a purely selfish system. Technically, we can define the Fear Factor Ψ as the inverse of the Price of Malice, i.e.,

$$\Psi(b) \;:=\; \frac{1}{PoM(b)}.$$

In other words, the Fear Factor Ψ quantifies how much the threat of a common enemy can unite selfish individuals, and to what degree the global social performance is improved.

In the virus inoculation game, the Fear Factor may be both negative and positive. What is interesting to note, however, is that this Fear Factor Ψ cannot be arbitrarily large, regardless of the number of Byzantine players b in the system. Instead, the Price of Malice can never drop below the constant $\frac{\sqrt{\pi}}{48}$ and hence, the Fear Factor is upper-bounded by $\Psi \leq \frac{48}{\sqrt{\pi}}$. That is, the social welfare or efficiency gained due to the Fear Factor cannot exceed a factor of $\Psi \leq \frac{48}{\sqrt{\pi}}$.

The existence of a Fear Factor has been documented in various economic and social models. By combining a game theoretic framework with the classic notion of Byzantine players from distributed computing and cryptography, our model allows for an *analytical quantification of a system's Fear Factor* Ψ from a computational point of view.

10.3 Stability Considerations

In the previous section, we have studied the degradation of the social welfare in a selfish system caused by Byzantine players. However, besides trying to reduce the optimality of certain outcomes of games, Byzantine players might also attack the *stability* of a system. In this section, we therefore continue our studies by capturing the amount of instability that can be caused by Byzantine players in an otherwise selfish system. Particularly, we are interested in the question, how many Byzantine players suffice in order to keep the system from stabilizing.

In the following, we generalize the model of Chapter 10.2 to *arbitrary* network graphs. We assume that the Byzantine players aim at destabilizing the system by repeatedly announcing to have changed from insecure to secure state and back in a worst-case fashion. Thereby, we consider an oblivious model where selfish nodes are not aware of the stability attack. We use the following definitions.

Definition 10.10 (b-Stable / b-Instable)**.** *We call a game b-stable if b Byzantine players cannot prevent the system from reaching a Nash equilibrium. Similarly, a game is called b-instable if b Byzantine players are sufficient such that no Nash equilibrium will ever be reached in case of oblivious selfish players.*

For the virus inoculation game, the following stability properties can be shown.

Theorem 10.12. *(i) Generally, the virus inoculation game is not 1-stable. (ii) For certain restricted classes of network graphs, the virus inoculation game is 1-stable. (iii) The virus inoculation game is always 2-instable.*

Proof. Claim (i): This claim already holds in simple graphs. Assume that n/L is an integer and that $L > 1$, and consider a one-dimensional chain of nodes $\{0, 1, ..., n-1\}$. Let the nodes $i \cdot n/L$ be secure, for $i \in \{0, 1, ..., L-1\}$. By Lemma 10.1, this situation constitutes a Nash equilibrium. Now assume that node n/L is Byzantine, and that it changes to the insecure state. Then, all other nodes $j \in \{1, 2, 3, ..., n/L-1, n/L+1, ..., 2n/L-1\}$ have an incentive to inoculate. However, once such a node j has become secure, node n/L can return to the secure state, yielding components of size smaller than n/L. Consequently, j is bound to become insecure again. These changes can be repeated forever.

Claim (ii): Interestingly, there are robust graphs where no single node can destabilize the system. To see this, consider a complete graph where each node is connected to all other nodes. From Lemma 10.1, it follows that in this network, all Nash equilibria have just one single attack component. Let $\mathcal{C}$ denote the set of nodes of this component, and let $\overline{\mathcal{C}} := V \setminus \mathcal{C}$ be the set of the remaining (secure) nodes. Also by Lemma 10.1, it holds that in any Nash equilibrium, the size of $\mathcal{C}$ is either n/L or $n/L - 1$. Moreover, observe that independently of which node is Byzantine and of how the Byzantine node acts, a situation will eventually be reached with the two components as described above. However, the system having converged to such a state, there exist only four possibilities: either the Byzantine node belongs to the node set $\mathcal{C}$ or to the node set $\overline{\mathcal{C}}$, and either $|\mathcal{C}| = n/L$ or $|\mathcal{C}| = n/L - 1$. It is run of the mill to verify that in all cases, a Byzantine node can enforce at most one additional change.

Claim (iii): We use the fact that in the virus inoculation game, a pure Nash equilibrium always exists, and that in the absence of Byzantine nodes, selfish nodes stabilize quickly [24]. Assume that the Byzantine nodes first act like selfish nodes until such a classic Nash equilibrium is reached. Now consider an arbitrary secure node $u_1 \in V$, and assume it is Byzantine. If u_1

becomes insecure, according to Lemma 10.1, an attack component $\mathcal{C}$ emerges which consists of n/L or more nodes. If $|\mathcal{C}| > n/L$, at least one node v in $\mathcal{C}$ has an incentive to change to a secure state. Let $\mathcal{C}'$ be the component of v when u_1 is secure, but not v. Assume that after v has changed, u_1 becomes secure again. There are two possibilities. If $|\mathcal{C}'| < n/L$, v will return to insecure state, and the changes can be repeated forever with only one Byzantine node. If $|\mathcal{C}'| = n/L$, a second Byzantine (previously insecure) node u_2 in $\mathcal{C}'$ can force v to become insecure again.

Finally, if $|\mathcal{C}| = n/L$, nodes are indifferent between becoming secure or not. Of course, however, another Byzantine node on the edge of $\mathcal{C}$ can cause endless changes also in this case. □

10.4 Concluding Remarks

This chapter has initiated the study of distributed systems consisting of both selfish and malicious players. Our framework can be applied not only to p2p systems but potentially many other economic and social systems. Using our models, we have derived bounds on the *Price of Malice* in oblivious and non-oblivious systems. Moreover, we have quantified and upper bounded the *Fear Factor*, which is the *gain* in system efficiency arising from the increased willingness of selfish individuals to cooperate caused by malicious players.

Several questions are left for future research. For example: what is the Price of Malice in a virus inoculation game on other topologies, e.g., on a hypercubic Pastry network? And what is the Price of Malice of other games, e.g., of a p2p *caching game* [63, 114]? It seems that while in certain selfish routing games where a single node can attract a lot of traffic by announcing short distances to all other nodes which results in a large Price of Malice, in congestion games the impact of Byzantine players may be much smaller. Another direction for future work is to study the *impact of knowledge* on the resulting Fear Factor in non-oblivious models. Specifically, one could assume that players are not only aware of the existence of Byzantine players, but also of their approximate whereabouts or their statistical distribution. Intuitively, such additional knowledge should decrease the selfish players' incentive for collaboration and thus lower the Fear Factor.

In Chapter 11, we will show how our framework can be adapted to study alternative phenomena in distributed systems. We will look at *social networks* and ask: how much damage can a virus entail in networks where players may care about their friends' welfare? We will see that the situation can only improve compared to purely selfish environments.

Chapter 11

Impact of Social Players

We now describe how to adapt the framework introduced in Chapter 10 for *social networks.* In particular, we investigate what happens in the virus inoculation game if users start caring about their friends' welfare.

Social networks have existed for thousands of years, but it was not until recently that researchers have started to gain scientific insights into phenomena like the *small world property*, which so far have only be known at an anecdotal level. The Internet has enabled people to connect with each other in new ways and to find friends sharing the same interests from all over the planet. A social network on the Internet can manifest itself in various forms. For instance, on *Facebook*, people maintain virtual references to their friends. The contacts stored in a user's Skype contact list, mobile phone, or email client form a social network as well. The analysis of such networks is an interesting endeavor, as they comprise many aspects of our society in general.

We believe that game theory can help to shed light on certain aspects of such systems. However, for the case of social networks, modeling players as completely self-interested may not be appropriate. It is likely that users act less selfishly towards their contacts than towards complete strangers. This chapter aims at taking such aspects into account. A framework is introduced which takes into consideration that players care about the well-being of their friends. In particular, we define the *Windfall of Friendship* (WoF) which captures to what extent the social welfare improves in social networks compared to purely selfish systems.

Social networks are not only attractive to their participants. It is well-known that the user profiles are an interesting data source for the PR industry to provide tailored advertisements. Moreover, the social network graph can also be exploited for attacks: for instance, email viruses have exploited the address books for propagating to the user's contacts, and worms spreading on mobile phone networks and over the Internet telephony tool Skype have been reported.

We will investigate the propagation of such viruses on social networks with the game introduced in Chapter 10. As expected, our analysis reveals that the players in this game always benefit from caring about the other participants in the social network rather than being selfish. Intriguingly, however, we find that the Windfall of Friendship does not increase monotonically with stronger relationships, i.e., with a larger *friendship factor* F. Despite of the phenomenon being an "ever-green" in political debates, to the best of our knowledge, this is the first framework to quantify this effect formally.

As another contribution, this chapter shows that computing the best and the worst friendship Nash equilibrium is **NP**-hard. In addition, simple networks such as cliques and stars are considered For example, we show that the Windfall of Friendship in cliques is at most 4/3; this is tight in the sense that there are problem instances where the situation can indeed improve this much. Moreover, we show that in star graphs, friendship can help to eliminate undesirable equilibria.

11.1 Framework

We consider again the virus inoculation game on an undirected network graph $G = (V, E)$ of $n = |V|$ players (or nodes) $p_i \in V$, for $i = 1, \ldots, n$, who are connected by a set of edges (or *links*) E. $a_i = 1$ denotes that player p_i is protected whereas for a player p_j willing to take the risk, $a_j = 0$. Assume that it costs C to inoculate, and that it costs L if a player gets infected. A player has the following expected cost:

Definition 11.1 (Actual Individual Cost). *The* (actual) individual cost *of a player p_i is defined as*

$$c_a(i, \vec{a}) = a_i \cdot C + (1 - a_i)L \cdot \frac{k_i}{n}$$

where k_i denotes the size of p_i's attack component. If p_i is inoculated, k_i will denote the attack component that would result if p_i became insecure. In the following, let $c_a^0(i, \vec{a})$ refer to the actual cost of an insecure and $c_a^1(i, \vec{a})$ to the cost of a secure node p_i.

We only consider pure Nash equilibria where it must hold for each player p_i, that given a strategy profile $\vec{a}$

$$\forall p_i \in V: \; c_a(i, \vec{a}) \leq c_a(i, (a_1, \ldots, a_i', \ldots, a_n)),$$

implying that player p_i cannot decrease her cost by choosing an alternative strategy a_i', given the other players' strategies. The total *social cost* of a game is the sum of the cost of all participants: $C_a(\vec{a}) = \sum_{p_i \in V} c_a(i, \vec{a})$. The *Price of Anarchy* (PoA) is the maximum ratio (over all problem instances I) between the worst Nash equilibrium cost divided by the social optimum, $PoA(I) = \max_{NE} C_{NE}(I)/C_{OPT}(I)$. In the remainder of this chapter, we only consider a limited region of the parameter space to avoid trivial cases: we will assume that $C \leq L$ and $C > L/n$. We make use of the following notation. For a player p_i, let $\Gamma(p_i)$ denote the set of neighbors of p_i. Moreover, let

$\Gamma_{sec}(p_i) \subseteq \Gamma(p_i)$ be the set of inoculated neighbors, and $\Gamma_{\overline{sec}}(p_i) = \Gamma(p_i) \setminus \Gamma_{sec}(p_i)$ be the remaining (insecure) neighbors.

We now introduce our model for social networks. We define a *Friendship Factor F* which captures the extent to which players care about their *friends*, i.e., about the players *adjacent* to them in the social network. More formally, F is the factor by which a player p_i takes the individual cost of her neighbors $\Gamma(p_i)$ into account when deciding for a strategy. F can assume any value between 0 and 1. $F = 0$ implies that the players do not consider their neighbors' cost at all, whereas $F = 1$ implies that a player values the well-being of her neighbors to the same extent as her own.

We distinguish between a player's *actual (individual) cost* and a player's *perceived cost.* A player's actual individual cost is the expected cost arising for each player defined in Definition 11.1 and we use it to compute a game's social cost. However, in our social network, the decisions of our players are steered by the players' *perceived* cost. Formally, the *perceived (individual) cost* is defined as follows.

Definition 11.2 (Perceived Individual Cost)**.** *The* perceived cost *of a player p_i is defined as*

$$c_p(i, \vec{a}) = c_a(i, \vec{a}) + F \cdot \sum_{p_j \in \Gamma(p_i)} c_a(j, \vec{a}).$$

In the following, we will again write $c_p^0(i, \vec{a})$ to denote the perceived cost of an insecure node p_i and $c_p^1(i, \vec{a})$ for the perceived cost of an inoculated node.

This definition entails a new equilibrium concept. We define a *friendship Nash equilibrium* (FNE) as a situation where no player can reduce her *perceived* cost by unilaterally changing her strategy given the strategies of the other players. Formally:

$$\forall p_i \in V : c_p(i, \vec{a}) \leq c_p(i, (a_1, \ldots, a_i', \ldots, a_n)).$$

Given this equilibrium concept, we can now define the *Windfall of Friendship Υ*.

Definition 11.3 (Windfall of Friendship (WoF) Υ)**.** *For a problem instance I, the Windfall of Friendship $\Upsilon(F, I)$ is the ratio of the (actual) social cost of the network's most expensive Nash equilibrium and the (actual) social cost of the worst friendship Nash equilibrium:*

$$\Upsilon(F, I) = \frac{\max_{NE} C_{NE}(I)}{\max_{BNE} C_{FNE}(F, I)}.$$

$\Upsilon(F) > 1$ implies the existence of a real windfall in the system, whereas $\Upsilon(F) < 1$ denotes that the social cost can become *greater* in social graphs than in purely selfish environments.

11.2 Windfall of Friendship

This section characterizes friendship Nash equilibria and derives general results on the Windfall of Friendship for the virus propagation game in social networks. It has been shown in [24] that in classic Nash equilibria ($F = 0$), an attack component can never consist of more than Cn/L insecure nodes. A similar characteristic also holds for FNEs. As every player cares about her neighbors, the maximal attack component size in which an insecure player p_i still does not inoculate depends on the number of p_i's insecure neighbors and the size of their attack components. Therefore, it differs from player to player. We have the following helper lemma.

Lemma 11.1. *The player p_i will inoculate if and only if the size of her attack component is*

$$k_i > \frac{Cn/L + F \cdot \sum_{p_j \in \Gamma_{\overline{sec}}(p_i)} k_j}{1 + F|\Gamma_{\overline{sec}}(p_i)|},$$

where the k_js are the attack component sizes of p_i's insecure neighbors if p_i is inoculated.

Proof. Player p_i will inoculate if and only if this choice lowers the perceived cost. By Definition 11.2, the perceived individual cost of an inoculated node are

$$c_p^1(i, \vec{a}) = C + F\left(|\Gamma_{sec}(p_i)|C + \sum_{p_j \in \Gamma_{\overline{sec}}(p_i)} L\frac{k_j}{n}\right),$$

and for an insecure node we have

$$c_p^0(i, \vec{a}) = L\frac{k_i}{n} + F\left(|\Gamma_{sec}(p_i)|C + |\Gamma_{\overline{sec}}(p_i)|L\frac{k_i}{n}\right).$$

For p_i to prefer to inoculate, it must hold that

$$c_p^0(i, \vec{a}) > c_p^1(i, \vec{a}) \Leftrightarrow$$

$$L\frac{k_i}{n} + F \cdot |\Gamma_{\overline{sec}}(p_i)|L\frac{k_i}{n} > C + F \cdot \sum_{p_j \in \Gamma_{\overline{sec}}(p_i)} L\frac{k_j}{n} \Leftrightarrow$$

$$L\frac{k_i}{n}(1 + F|\Gamma_{\overline{sec}}(p_i)|) > C + \frac{FL}{n} \cdot \sum_{p_j \in \Gamma_{\overline{sec}}(p_i)} k_j \Leftrightarrow$$

$$k_i(1 + F|\Gamma_{\overline{sec}}(p_i)|) > Cn/L + F \cdot \sum_{p_j \in \Gamma_{\overline{sec}}(p_i)} k_j \Leftrightarrow$$

$$k_i > \frac{Cn/L + F \cdot \sum_{p_j \in \Gamma_{\overline{sec}}(p_i)} k_j}{1 + F|\Gamma_{\overline{sec}}(p_i)|}.$$

□

A pivotal question is of course whether social networks where players care about their friends yield better equilibria than selfish environments. The following theorem answers this questions affirmatively: the worst FNE never costs more than the worst NE.

Theorem 11.2. *For all instances of the virus inoculation game and* $0 \leq F \leq 1$*, it holds that*

$$n \geq PoA \geq \Upsilon(F) \geq 1.$$

Proof. The proof idea for $\Upsilon(F) \geq 1$ is the following: for an instance I we consider an arbitrary FNE with $F > 0$. Given this equilibrium, we prove the existence of a NE with larger social cost. Let α be any (e.g., the worst) FNE in the social model. If α is also a NE in the same instance with $F = 0$ then we are done. Otherwise there is at least one player p_i that prefers to change her strategy. Assume p_i is insecure but favors inoculation. Therefore p_i's attack component has on the one hand to be of size at least $k_i' > Cn/L$ [24] and on the other hand of size at most $k_i'' = (Cn/L + F \cdot \sum_{p_j \in \Gamma_{\overline{sec}}(p_i)} k_j)/(1 + F|\Gamma_{\overline{sec}}(p_i)|) \leq (Cn/L + F|\Gamma_{\overline{sec}}(p_i)|(k_i'' - 1))/(1 + F|\Gamma_{\overline{sec}}(p_i)|) \Leftrightarrow k_i'' \leq Cn/L - F|\Gamma_{\overline{sec}}(p_i)|$ (cf Lemma 11.1). This is impossible and yields a contradiction to the assumption that in the selfish network, an additional player wants to inoculate.

It remains to study the case where p_i is secure in the FNE but prefers to be insecure in the NE. Observe that, since every player has the same preference on the attack component's size when $F = 0$, a newly insecure player cannot trigger other players to inoculate. Furthermore, only the players inside p_i's attack component are affected by this change. The total cost of this attack component increases by at least

$$\begin{aligned} x &= \underbrace{\frac{k_i}{n}L - C}_{p_i} + \underbrace{\sum_{p_j \in \Gamma_{\overline{sec}}(p_i)} \left(\frac{k_i}{n}L - \frac{k_j}{n}L\right)}_{p_i\text{'s insecure neighbors}} \\ &= \frac{k_i}{n}L - C + \frac{L}{n}(|\Gamma_{\overline{sec}}(p_i)|k_i - \sum_{p_j \in \Gamma_{\overline{sec}}(p_i)} k_j). \end{aligned}$$

Lemma 11.1 guarantees that $\sum_{p_j \in \Gamma_{\overline{sec}}(p_i)} k_j \leq (k_i(1 + F|\Gamma_{\overline{sec}}(p_i)|) - Cn/L)/F$. This results in

$$\begin{aligned} x &\geq \frac{k_i L}{n} - C + \frac{L}{n}\left[|\Gamma_{\overline{sec}}(p_i)|k_i - \frac{k_i(1 + F|\Gamma_{\overline{sec}}(p_i)|) - Cn/L}{F}\right] \\ &= \frac{k_i L}{n}(1 - \frac{1}{F}) - C(1 - \frac{1}{F}) > 0, \end{aligned}$$

since a player only gives up her protection if $C > k_i L/n$. If even more players are unhappy with their situation and become vulnerable, the cost for the NE increases more. In conclusion, there exists a NE for every FNE with $F \geq 0$ for the same instance which is at least as expensive.

The upper bound for the WoF, i.e., PoA $\geq \Upsilon(F)$, follows directly from the definitions: while the PoA is the ratio of the NE's social cost divided by the social optimum, $\Upsilon(F)$ is the ratio between the cost of the NE and the FNE. As the FNE's cost must be at least as large as the social optimum cost, and since $n \geq$ PoA [24], the claim follows. □

The above result leads to the question of whether the Windfall of Friendship grows monotonically with stronger social ties, i.e., with larger friendship factors F. Intriguingly, this is not the case.

Theorem 11.3. *For all network sizes $n > 7$, there exist game instances for which $\Upsilon(F)$ does not grow monotonically in F.*

Proof. We give a counter example for the star graph S_n which has one center node and $n-1$ boundary nodes. Consider two friendship factors, F_l and F_s where $F_l > F_s$. We show that for the large friendship factor F_l, there exists a FNE, FNE_1, where only the center node and one boundary node remain insecure. For the same setting but with a small friendship factor F_s, at least two boundary nodes will remain insecure, which will trigger the center node to inoculate, yielding a FNE, FNE_2, where only the center node is secure.

Consider FNE_1 first. Let c be the insecure center node, let b_1 be the insecure boundary node, and let b_2 be a secure boundary node. In order for FNE_1 to constitute a Nash equilibrium, the following conditions must hold:

$$\begin{aligned}
\text{node } c: &\quad \frac{2L}{n} + \frac{2F_lL}{n} < C + \frac{F_lL}{n} \\
\text{node } b_1: &\quad \frac{2L}{n} + \frac{2F_lL}{n} < C + \frac{F_lL}{n} \\
\text{node } b_2: &\quad C + \frac{2F_lL}{n} < \frac{3L}{n} + \frac{3F_lL}{n}
\end{aligned}$$

For FNE_2, let c be the insecure center node, let b_1 be one of the two insecure boundary nodes, and let b_2 be a secure boundary node. In order for the boundary nodes to be happy with their situation but for the center node to prefer to inoculate, it must hold that:

$$\begin{aligned}
\text{node } c: &\quad C + \frac{2F_sL}{n} < \frac{3L}{n} + \frac{6F_sL}{n} \\
\text{node } b_1: &\quad \frac{3L}{n} + \frac{3F_sL}{n} < C + \frac{2F_sL}{n} \\
\text{node } b_2: &\quad C + \frac{3F_sL}{n} < \frac{4L}{n} + \frac{4F_sL}{n}
\end{aligned}$$

Now choose $C := 5L/(2n) + F_lL/n$. This yields the following conditions: $F_l > F_s + 1/2$, $F_l < F_s + 3/2$, and $F_l < 4F_s + 1/2$. These conditions are easily fulfilled, e.g., for $F_l = 3/4$ and $F_s = 1/8$. Observe that the social cost of the first FNE (for F_l) is $Cost(S_n, \vec{a}_{FNE1}) = (n-2)C + 4L/n$, whereas for the second FNE (for F_s) $Cost(S_n, \vec{a}_{FNE2}) = C + (n-1)L/n$. Thus, $Cost(S_n, \vec{a}_{FNE1}) - Cost(S_n, \vec{a}_{FNE2}) = (n-3)C - (n+3)L/n > 0$ as we have chosen $C > 5L/(2n)$ and as, due to our assumption, $n > 7$. This concludes the proof. □

Reasoning about best and worst Nash equilibria raises the question how difficult it is to compute such equlibria. We can generalize the proof given in [24] and show that computing the most economical and the most expensive FNE is hard for any friendship factor.

Theorem 11.4. *Computing the best and the worst pure friendship Nash equilibrium is* **NP**-*complete for any* $0 \leq F \leq 1$.

Proof. We prove this theorem by a reduction from two **NP**-hard problems, Vertex Cover [125] and Independent Dominating Set [101]. Concretely, for the decision version of the problem, we show that answering the question of whether there exists a FNE with cost less than k, or more than k respectively, is at least as hard as solving vertex cover or independent dominating set. Note that verifying whether a proposed solution is correct can be done in polynomial time, hence the problems are indeed in **NP**.

Fix some graph $G = (V, E)$ and set $C = 1$ and $L = n/1.5$. We show first that the following two conditions are necessary and sufficient for a FNE: (a) all neighbors of an insecure node are secure, and (b) every inoculated node has at least one insecure neighbor.

Due to our assumption that $C > L/n$, Condition (b) holds. To see that Condition (a) holds as well, assume that there is an attack component of size at least two. An insecure node p_i in this attack component bears the cost $\frac{k_i}{n}L + F(|\Gamma_{sec}(p_i)|C + |\Gamma_{\overline{sec}}(p_i)|\frac{k_i}{n}L)$. Changing the strategy reduces the cost by at least $\Delta_i = \frac{k_i}{n}L + F|\Gamma_{\overline{sec}}(p_i)|\frac{k_i}{n}L - C - F|\Gamma_{\overline{sec}}(p_i)|\frac{k_i-1}{n}L = \frac{k_i}{n}L + F|\Gamma_{\overline{sec}}(p_i)|\frac{1}{n}L - C$. By our assumption that $k_i \geq 2$, and hence $|\Gamma_{\overline{sec}}(p_i)| \geq 1$, it holds that $\Delta_i > 0$, resulting in p_i becoming secure. Therefore, Condition (a) holds as well. For the opposite direction assume that an insecure node wants to change the strategy even though (a) and (b) are true. This is impossible because in this case (b) would be violated. An inoculated node would destroy (a) by adopting another strategy. Thus (a) and (b) are sufficient for a FNE.

We now argue that G has a vertex cover of size k if and only if the virus game has a FNE with k or fewer secure nodes, or equivalently an equilibrium with social cost at most $Ck + (n-k)L/n$, as each insecure node must be in a component of size 1 and contributes exactly L/n expected cost. Given a minimal vertex cover $V' \subseteq V$, observe that installing the software on all nodes in V' satisfies Condition (a) because V' is a vertex cover and (b) because V' is minimal. Conversely, if V' is the set of secure nodes in a FNE, then V' is a vertex cover by Condition (a) which is minimal by Condition (b).

For the worst FNE, we consider an instance of the independent dominating set problem. Given an independent dominating set $V' \subseteq V$, installing the software on all nodes except the nodes in V' satisfies Condition (a) because V' is independent and (b) because V' is a dominating set. Conversely, the insecure nodes in any FNE are independent by Condition (a) and dominating by Condition (b). This shows that G has an independent dominating set of size at most k iff it has a FNE with at least $n-k$ secure nodes. □

11.3 Clique and Star

We study the Windfall of Friendship for two concrete topologies, namely the *complete graph* K_n and the *star graph* S_n.

11.3.1 Clique

We consider a most simple topology, namely the complete graph, or *clique*, K_n where all players are connected to each other. First assume a classic setting where nodes do not care about their neighbors ($F=0$). We have the following result:

Lemma 11.5. *In the clique K_n, there are two Nash equilibria with total cost:*

NE_1: $Cost(K_n, \vec{a}_{NE1}) = (n - \lceil Cn/L \rceil - 1)C + (\lceil Cn/L \rceil - 1)^2 L/n$

NE_2: $Cost(K_n, \vec{a}_{NE2}) = (n - \lfloor Cn/L \rfloor)C + (\lfloor Cn/L \rfloor)^2 L/n$

If $\lceil Cn/L - 1 \rceil = \lfloor Cn/L \rfloor$, there is only one Nash equilibrium.

Proof. Let $\vec{a}$ be a NE. Consider an inoculated node p_i and an insecure node p_j, and hence $c_a(i,\vec{a}) = C$ and $c_a(j,\vec{a}) = L\frac{k_j}{n}$, where k_j is the total number of insecure nodes in K_n. In order for p_i to remain inoculated, it must hold that $C \le (k_j+1)L/n$, so $k_j \ge \lceil Cn/L - 1 \rceil$; for p_j to remain insecure, it holds that $k_j L/n \le C$, so $k_j \le \lfloor Cn/L \rfloor$. As the total social cost in K_n is given by $Cost(K_n,\vec{a}) = (n-k_j)C + k_j^2 L/n$, the claim follows. □

Observe that the equilibrium size of the attack component is roughly twice the size of the attack component of the social optimum, as $Cost(K_n,\vec{a}) = (n-k_j)C + k_j^2 L/n$ is minimized for $k_j = Cn/2L$.

Lemma 11.6. *In the social optimum for K_n, the size of the attack component is either $\lfloor \frac{1}{2} Cn/L \rfloor$ or $\lceil \frac{1}{2} Cn/L \rceil$, yielding a total social cost of*

$$Cost(K_n, \vec{a}_{OPT}) = (n - \lfloor \frac{1}{2} Cn/L \rfloor)C + (\lfloor \frac{1}{2} Cn/L \rfloor)^2 \frac{L}{n}$$

or

$$Cost(K_n, \vec{a}_{OPT}) = (n - \lceil \frac{1}{2} Cn/L \rceil)C + (\lceil \frac{1}{2} Cn/L \rceil)^2 \frac{L}{n},$$

respectively.

In order to compute the Windfall of Friendship, the FNEs in social networks have to be identified.

Lemma 11.7. *In K_n, there are two FNEs with total cost:*

$$FNE_1\text{: } Cost(K_n, \vec{a}_{FNE1}) = (n - \lceil \tfrac{Cn/L-1}{1+F} \rceil)C + (\lceil \tfrac{Cn/L-1}{1+F} \rceil)^2 L/n$$

$$FNE_2\text{: } Cost(K_n, \vec{a}_{FNE2}) = (n - \lfloor \tfrac{Cn/L+F}{1+F} \rfloor)C + (\lfloor \tfrac{Cn/L+F}{1+F} \rfloor)^2 L/n$$

If $\lceil (Cn/L - 1)/(1 + F) \rceil = \lfloor (Cn/L + F)/(1 + F) \rfloor$, there is only one FNE.

Proof. According to Lemma 11.1, in a FNE, a node p_i remains secure if otherwise the component had size at least $k_i = k_j + 1 \geq (Cn/L + Fk_j^2)/(1 + Fk_j)$, where k_j is the number of insecure nodes. This implies that $k_j \geq \lceil (Cn/L - 1)/(1 + F) \rceil$. Dually, for an insecure node p_j it holds that $k_j \leq (Cn/L + F(k_j - 1)^2)/(1 + F(k_j - 1))$ and therefore $k_j \leq \lfloor (Cn/L + F)/(1 + F) \rfloor$. Given these bounds on the total number of insecure nodes in a FNE, the social cost can be obtained by substituting k_j in $Cost(K_n, \vec{a}) = (n - k_j)C + k_j^2 L/n$. As the difference between the upper and the lower bound for k_j is at most 1, there are at most two equilibria and the claim follows. □

Given the characteristics of the different equilibria, we have the following theorem.

Theorem 11.8. *In K_n, the Windfall of Friendship is at most $\Upsilon(F) = 4/3$ for arbitrary network sizes. This is tight in the sense that there are indeed instances where the worst FNE is a factor $4/3$ better than the worst NE.*

Proof. Upper Bound. We first derive the upper bound on $\Upsilon(F)$.

$$\begin{aligned} \Upsilon(F) &= \frac{Cost(K_n, \vec{a}_{NE})}{Cost(K_n, \vec{a}_{FNE})} \\ &\leq \frac{Cost(K_n, \vec{a}_{NE})}{Cost(K_n, \vec{a}_{OPT})} \\ &\leq \frac{(n - \lceil Cn/L - 1 \rceil)C + (\lfloor Cn/L \rfloor)^2 \frac{L}{n}}{(n - \frac{1}{2}Cn/L)C + (\frac{1}{2}Cn/L)^2 \frac{L}{n}} \end{aligned}$$

as the optimal social cost (cf Lemma 11.6) is smaller or equal to the social cost of any FNE. Simplifying this expression yields

$$\Upsilon(F) \leq \frac{n(1 - C/L)C + C^2 n/L}{n(1 - \frac{1}{2}C/L)C + \frac{1}{4}C^2 n/L} = \frac{1}{1 - \frac{1}{4}C/L}.$$

This term is maximized for $L = C$, implying that $\Upsilon(F) \leq 4/3$, for arbitrary n.

Lower Bound. We now show that the ratio between the equilibria cost reaches $4/3$. There exists exactly one social optimum of cost $Ln/2 + (n/2)^2 L/n = 3nL/4$ for even n and $C = L$ by Lemma 11.6. For $F = 1$ this is also the only friendship Nash equilibrium due to Lemma 11.7. In the selfish game however the Nash equilibrium has fewer inoculated nodes and is of cost nL

(see Lemma 11.5). Since these are the only equilibria they constitute the worst equilibria and the ratio becomes

$$\Upsilon(F) = \frac{Cost(K_n, \vec{a}_{NE})}{Cost(K_n, \vec{a}_{FNE})} = \frac{nL}{3/4nL} = 4/3.$$

□

To conclude our analysis of K_n, observe that FNEs always exist in cliques, and that best response strategies where one node at a time is given the chance to change its strategy quickly results in such an equilibrium.

11.3.2 Star

While the analysis of the K_n was simple, it turns out that already slightly more sophisticated graphs are challenging. In the following, the Windfall of Friendship is investigated in star graphs S_n. Note that in S_n, the social welfare is maximized if the center node inoculates and all other nodes do not. The total inoculation cost then is C and the attack components are all of size 1, yielding a total social cost of $Cost(S_n, \vec{a}_{OPT}) = C + (n-1)L/n$.

Lemma 11.9. *In the social optimum of the star graph S_n, only the center node is inoculated. The social cost is $Cost(S_n, \vec{a}_{OPT}) = C + (n-1)L/n$.*

The situation where only the center node is inoculated also constitutes a NE. However, there are more Nash equilibria.

Lemma 11.10. *In the star graph S_n, there are three NEs with total cost:*

NE$_1$: $Cost(S_n, \vec{a}_{NE1}) = C + (n-1)L/n$

NE$_2$: $Cost(S_n, \vec{a}_{NE2}) = (n - \lceil Cn/L \rceil) + 1C + (\lceil Cn/L \rceil - 1)^2 L/n$

NE$_3$: $Cost(S_n, \vec{a}_{NE3}) = (n - \lfloor Cn/L \rfloor)C + (\lfloor Cn/L \rfloor)^2 L/n$

If $Cn/L \notin \mathbb{N}$, only two equilibria exist.

Proof. If the center node is the only secure node, changing to insecure state costs L but saves only C. When a boundary node becomes secure, its cost changes from L/n to C. These changes are unprofitable, and the social cost of this NE is $Cost(S_n, \vec{a}_{NE1}) = C + (n-1)L/n$.

For the other NEs the center node is not inoculated. Let the number of insecure boundary nodes be n_0. In order for a secure node to remain secure, it must hold that $C \leq (n_0 + 2)L/n$, and hence $n_0 \geq \lceil Cn/L - 2 \rceil$. In order for an insecure node to remain insecure, it must hold that $(1 + n_0)L/n \leq C$, so $n_0 \leq \lfloor Cn/L - 1 \rfloor$. Therefore, we can conclude that there are at most two Nash equilibria, one with $\lceil Cn/L - 1 \rceil$ and one with $\lfloor Cn/L \rfloor$ many insecure nodes. The total social cost follows by substituting n_0 in the total social cost function. Finally, observe that if $Cn/L \in \mathbb{N}$ and $Cn/L > 3$, all three equilibria exist in parallel. □

Let us consider the social scenario again.

Lemma 11.11. *In S_n, there are at most three FNEs with total cost:*

FNE$_1$*:* $Cost(S_n, \vec{a}_{FNE1}) = C + (n-1)L/n$

FNE$_2$*:* $Cost(S_n, \vec{a}_{FNE2}) = (n - \lceil Cn/L - 1 - F \rceil)C + (\lceil Cn/L - 1 - F \rceil)^2 L/n$

FNE$_3$*:* $Cost(S_n, \vec{a}_{FNE3}) = (n - \lfloor Cn/L - F \rfloor)C + (\lfloor Cn/L - F \rfloor)^2 L/n$

If $Cn/L - F \notin \mathbb{N}$*, at most 2 friendship Nash equilibria exist.*

Proof. First, observe that having only an inoculated center node constitutes a FNE. In order for the center node to remain inoculated, it must hold that $C + F(n-1)L\frac{1}{n} \leq nL/n + F(n-1)L\frac{n}{n} = L + F(n-1)L$. All boundary nodes remain insecure as long as $L/n + FC \leq C + FC \Leftrightarrow L/n \leq C$. These conditions are always true, and we have $Cost(S_n, \vec{a}_{FNE1}) = C + (n-1)L/n$.

If the center node is not inoculated, we have n_0 insecure and $n - n_0 - 1$ inoculated boundary nodes. In order for a secure boundary node to remain secure, it is necessary that $C + F\frac{n_0+1}{n}L \leq \frac{n_0+2}{n}L + F\frac{n_0+2}{n}L$, so $n_0 \geq \lceil Cn/L - 2 - F \rceil$. For an insecure boundary node, it must hold that $\frac{n_0+1}{n}L + F\frac{n_0+1}{n}L \leq C + F\frac{n_0}{n}L$, so $n_0 \leq \lfloor Cn/L - 1 - F \rfloor$. The claim follows by substitution. □

Note that it is indeed possible that the star graph has three FNEs in parallel. As an example, consider the star graph S_9 with $C = 1, L = 4$, and $F = 0.25$. Moreover, observe that there are instances where FNE$_1$ is the only friendship Nash equilibrium. We already made use of this phenomenon in Chapter 11.2 to show that $\Upsilon(F)$ is not monotonically increasing in F. The next lemma states under which circumstances this is the case.

Lemma 11.12. *In S_n, there is a unique FNE equivalent to the social optimum if and only if*

$$\lfloor Cn/L - F \rfloor - \lfloor \frac{1}{2F}(\sqrt{1 - 4F(1 - Cn/L)} - 1) \rfloor - 2 \geq 0$$

Proof. S_n has only one FNE if every (insecure) boundary node is content with its chosen strategy but the insecure center node would rather inoculate. In order for an insecure boundary node to remain insecure we have $n_0 \leq \lfloor Cn/L - 1 - F \rfloor$ and the insecure center node wants to inoculate if and only if $C + F(n - n_0 - 1)C + Fn_0\frac{1}{n}L < (n_0+1)\frac{L}{n} + F(n - n_0 - 1)C + Fn_0\frac{n_0+1}{n}L \Leftrightarrow Fn_0^2 + n_0 + 1 - Cn/L > 0 \Leftrightarrow n_0 \geq \lfloor \frac{1}{2F}(\sqrt{1 - 4F(1 - Cn/L)} - 1) + 1 \rfloor$. Therefore there is only one FNE iff there exists an integer n_0 such that $\lfloor \frac{1}{2F}(\sqrt{1 - 4F(1 - Cn/L)} - 1) + 1 \rfloor \leq n_0 \leq \lfloor Cn/L - 1 - F \rfloor$. □

Given the characterization of the various equilibria, the Windfall of Friendship can be computed.

Theorem 11.13. *If* $\lfloor Cn/L-1-F\rfloor-\lfloor\frac{1}{2F}(\sqrt{1-4F(1-Cn/L)}-1)+1\rfloor\geq 0$, *the Windfall of Friendship is*

$$\Upsilon(F)\geq\frac{(n-2)C+L/n}{C+(n-1)/nL}.$$

Otherwise, $\Upsilon(F)\leq\frac{n+1}{n-3}$.

Proof. According to Lemma 11.12, if $\lfloor Cn/L-F\rfloor-\lfloor\frac{1}{2F}(\sqrt{1-4F(1-Cn/L)}-1)\rfloor-2\geq 0$, the FNE is unique and hence equivalent to the social optimum. On the other hand, observe that there always exist expensive NEs where the center node is not inoculated. Hence, we have

$$\begin{aligned}\Upsilon(F) &= \frac{Cost(S_n,\vec{a}_{NE})}{Cost(S_n,\vec{a}_{FNE})}=\frac{Cost(S_n,\vec{a}_{NE})}{Cost(S_n,\vec{a}_{OPT})}\\ &\geq \frac{(n-\lfloor Cn/L-1\rfloor)C+(\lceil Cn/L\rceil-1)^2L/n}{C+(n-1)L/n}\\ &\geq \frac{C(n-2)+L/n}{C+(n-1)L/n}\ .\end{aligned}$$

Otherwise, i.e., if there exist FNEs with an insecure center node, an upper bound for the WoF can be computed:

$$\begin{aligned}\Upsilon(F) &= \frac{Cost(S_n,\vec{a}_{NE})}{Cost(S_n,\vec{a}_{FNE})}\\ &\leq \frac{(n-\lceil Cn/L-1\rceil)C+(\lfloor Cn/L\rfloor)^2L/n}{(n-\lfloor Cn/L-F\rfloor)C+(\lceil Cn/L-1-F\rceil)^2L/n}\\ &\leq \frac{(n+1)C}{nC+FC-2C(1+F)+(1+F)^2L/n}\\ &< \frac{(n+1)C}{C(n+F-2(1+F))}\quad<\quad\frac{n+1}{n-3}\ .\end{aligned}$$

□

Theorem 11.13 reveals that caring about the cost incurred by friends is particularly helpful to reach more desirable equilibria. In large star networks, the social welfare can be much higher than in Nash equilibria: in particular, the Windfall of Friendship can increase linearly in n, and hence indeed be asymptotically as large as the Price of Anarchy. However, if $\lfloor Cn/L-1-F\rfloor-\lfloor\frac{1}{2F}(\sqrt{1-4F(1-Cn/L)}-1)+1\rfloor\geq 0$ does not hold, social networks do not perform much better than purely selfish systems: the maximal gain is constant.

Finally observe that also in S_n, FNEs always exist and can be computed efficiently (in linear time) by a best response strategy.

11.4 Concluding Remarks

This chapter has studied viruses propagating along links of social networks, e.g., along contact lists of Skype users. Also in this context, our framework introduced in Chapter 10 helps to formally describe interesting phenomena. We have shown that the virus cannot do more harm than in selfish environments, and we have observed that the social welfare does not grow monotonically with stronger social ties.

We believe that our work opens many possibilities for future research. First, the analysis of the Windfall of Friendship needs to be continued for other graphs, especially for graphs incorporating more topological properties of real networks, like random graphs or Kleinberg graphs which exhibit the small world property. We have restricted ourselves to players who only care for their direct neighbors. Investigating the effects of considering players over multiple hops (maybe to a lesser extent) or letting the metric distance between two players decide about their degree of friendship, i.e., porting the virus inoculation game into the Euclidean space, are interesting research directions.

Chapter 12

Attacks Case Study: Kad

In order to complement our theoretic results given in Chapter 10 on the effects of malicious players in p2p systems, in the following, as a case study, we evaluate the feasibility of various malicious attacks in the Kad network—the most widely deployed structured p2p system with more than a million simultaneous users [223]. We find that while the Kad network functions reliably under normal operation, it has several critical vulnerabilities, despite ongoing efforts on the developers' part to prevent fraudulent and destructive use.

We describe several protocol exploits which prevent peers from accessing particular files in the system. In order to obstruct access to specific files, file requests can be hijacked, and subsequently, arbitrary information can be returned instead of the actual data. Alternatively, we show that publishing peers can be overwhelmed with bogus information such that pointers to the original files can no longer be accessed. Moreover, it is even possible to *eclipse* certain peers, i.e., to fill up their routing tables with information about malicious peers, which can subsequently intercept all messages. We will also briefly discuss how our network poisoning attacks can be used to harm machines *outside* the Kad network, e.g. web servers, by tricking the peers into performing a Kad-steered distributed denial of service (DDoS) attack. It is virtually impossible to determine the true culprit in this scenario, as the initiating peer does not take part in the attack, which makes this kind of vulnerability appealing to malicious peers.

All our attacks have been tested on the real Kad network using a modified C++ eMule client. Already with three attackers, virtually no peer in the system was able to find content associated with any given keyword for several hours, which demonstrates that with moderate computational resources, access to any targeted content can be undermined easily.

12.1 Kad Background

The Kad network is a DHT-based p2p network that implements the *Kademlia protocol* [161]. Access to the Kad network is provided through the *eMule*[1] client, which also uses the server-based *eDonkey*[2] network.

Each peer in the Kad network has a 128-bit identifier (ID) which is normally created by a random generator. This ID is stored at the peer even after it has left the network and is re-used once the peer returns. Routing in the network is performed using these identifiers and the XOR metric, which defines the distance between two identifiers as the bitwise exclusive or (XOR) of these identifiers interpreted as an integer. For all $i \in [0, 127]$, every peer stores the addresses of a few other peers whose distance to its own ID is between 2^i and 2^{i+1}, resulting in a connected network whose diameter is logarithmically bounded in the number of peers. For each of these *contacts* in the routing table, a Kad ID, an IP address, and a port is stored.

The publish and retrieval mechanisms work roughly as follows. Each keyword, i.e., a word in a file name, and the file itself, are hashed, and information about the keywords, its associated file, and the address of the owner is published in the network, i.e., this information is stored at the peers in the DHT whose identifers are closest to the respective hash values. If a peer wants to download a file with a certain name (a particular sequence of keywords), it first queries the peer whose identifier is closest to the hash of the *first* of the specified keywords, and this peer returns the information of all files whose file names contain all the given keywords, and also the corresponding file hashes. The requesting peer p_1 can then download the desired file by querying the peer p_2 whose identifier is closest to the file hash, as p_2 keeps track of all the peers in the network that actually own a copy of the file.

12.2 Attacks and Measurements

This section presents three different attacks which limit the Kad users' access to a given file f. In a *node insertion attack*, an attacking peer seeks to attract search requests for f, which are answered with bogus information. Alternatively, access to f can be denied by filling up the index tables of other peers publishing information about f (*publish attack*). Finally, we describe how an attacker can *eclipse* an arbitrary peer: by controlling all the peer's incoming and outgoing traffic, the attacker can prevent a peer from either publishing information about f or from accessing it.

12.2.1 Node Insertion Attack

By performing a *node insertion attack*, it is possible to corrupt the network by spreading polluted information, e.g., about the list of sources, keywords,

[1]See http://www.emule-project.net/.

[2]See http://en.wikipedia.org/wiki/EDonkey_network/.

or comments. We have implemented the attacks for *keywords*, that is, a search for the attacked keyword will not give the correct results, but instead arbitrary data chosen by the attacker is returned.

For this attack to work, we have to ensure that the search requests for the specific keyword are routed to the attacking peer rather than to the peers storing the original information. This can be achieved as follows. In the Kad network, a peer normally creates its ID using a random number generator; however, any alternative mechanism will work as well, as there is no verification of a peer's ID. In our modified eMule client, it is possible to select the peer's Kad ID manually. Thus, an attacker can choose its ID such that it matches the hash value of the targeted keyword. Consequently, the peer will become the node closest to this ID and will receive all the corresponding search requests. The nodes storing the correct files typically have a larger distance to the keyword's ID than the attacker, as the probability for a peer to have a random ID that perfectly matches the 128-bit keyword ID is negligible.

In order to guarantee that peers looking for a certain keyword only receive faked results, the attacker must provide enough result tuples, as the eMule client terminates the search after having gathered 300 tuples. The attacker further has to include the keywords received from a peer in the filenames, otherwise the replies are not accepted. In our attacks, we use filenames that contain a unique number, the message "File removed from Kad!", and the keywords. Unique file hashes are needed such that the 300 tuples are not displayed as one tuple in eMule's search window.

We frequently observed that eMule sends search requests not only to the closest peer, even though this peer provided enough answers. This can be explained by the delay caused when transmitting the 300 search results from the closest peer. eMule will send another request to the second closest peer before all of the results are received from the closest one. This of course may harm the effectiveness of the attack, and hence it is beneficial to gain control over the second, third, etc. closest IDs as well by means of additional attackers. These attackers behave exactly the same way: all requests are answered by supplying 300 faked tuples.

Figure 12.1 depicts the traces obtained during two week-long node insertion attacks performed using our modified eMule client on the keyword "Simpsons". Note that this attack influences all queries in the entire Kad network not only for the search term "Simpsons", but also all other queries starting with the term "Simpsons" such as "Simpsons Movie" or "Simpsons Soundtrack" etc. are affected automatically.

In the first trace, only one attacker whose ID exactly matches the hash of the keyword infiltrated the network. We used another client to search for the term "Simpsons" once a minute and examined the returned results. Since a single attacker is not sufficient, as mentioned before, the attack is moderately successful in that only approximately 40% of the returned results originated from the attacker. What is more, every single query returned at least some results that are not faked. Further experiments showed that using two attackers instead of one does not increase the success rate substantially, but three attackers is already enough to hijack virtually all requests. The

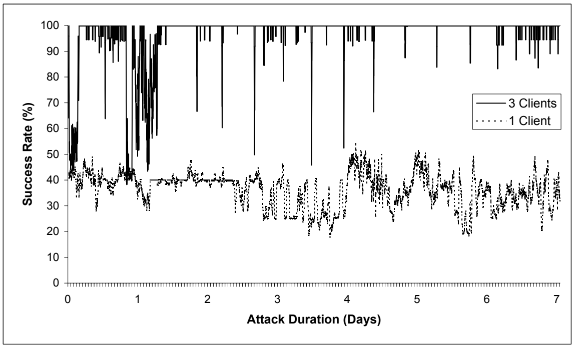


Figure 12.1: Percentage of successfully hijacked keyword requests in a node insertion attack for 1 and 3 attackers during a time period of one week.

second trace shows the success rate of the node insertion attack using three attackers. On average, more than 95% of all returned tuples were faked, and every batch of tuples contained at least some bogus data created by the attackers. The plot shows that there are sudden drops of the success rate once in a while. An explanation for this behavior is that peers join and leave the network at a high rate, resulting in inaccurate routing tables. Consequently, a lookup request can be routed to a peer that still stores results for this request and does not know about our attacking peers yet.

The attack was repeated at other times using different keywords. All our other experiment resulted in a similar picture and confirmed our findings made with the "Simpsons" keyword. Our attacking peers received roughly 8 requests per minute from other peers in the network during the experiments. As expected, the peer having the closest ID received the most requests at a rate of roughly 4 requests per minute.

12.2.2 Publish Attack

In contrast to the node insertion attack, which forces the search requests to be routed to the attacker, the publish attack directly attacks the peers closest to the ID of the attacked keyword, comment, or source entry. The index tables stored by the peers in the Kad network have a limited length; for instance, the keyword table can store up to 50,000 entries for a specific ID. Moreover, a peer will never return more than 300 result tuples per request, giving priority to the latest additions to the index table. This makes it possible to replace the original information by filling up the tables of the corresponding peers with poisoned entries. Thus, an attacker seeks to publish a large amount of information on these peers. Once the index tables of the attacked peers are full, they will not accept any publish requests by other peers anymore. Therefore, the attacked peers will only return our poisoned entries instead of the original information. Since every entry has an expiration time (24 hours for keyword and comment entries, and 5 hours for source entries), the clients have to be re-attacked periodically in order to achieve a constant fraction of poisoned entries. In addition, an attacker has to take into consideration the newly joining peers in the network; if they have an ID close to the one attacked, their tables also have to be filled.

We have implemented the publish attack for keyword entries as well, again by modifying the original eMule application. An existing timer method is used to run the attack every 10 minutes. In the first phase the 12 peers closest to the targeted ID are located, using eMule's search mechanism. In each run, only peers are selected that have not been attacked before or which need to be re-attacked due to the expiration of the poisoned entries. In the second phase, all the peers found in the first phase are attacked, beginning with the closest peer. To guarantee a full cache list, 50,000 poisoned entries are sent divided into 250 packets containing 200 entries each. In order to prevent overloading the attacked client, the sending rate was limited to 5 packets per second. Every entry consists of a unique hash value and filename as in the node insertion attack. Since these entries should match all search requests containing the attacked keyword, it is necessary to include all additional relevant keywords (e.g., song titles for an interpreter, year and language for a film title) in the filename; otherwise, all the lookups with additional keywords would not receive the poisoned entries. In the node insertion attack, this problem does not occur as the additional keywords are obtained from every search request and can directly be appended to the filename to match the request. The success of each run is measured with the load value sent in every response to a publish packet. This value should increase with every poisoned packet sent, from a starting level of about 10 - 20% to 100% when the attack is finished.

In comparison to the node insertion attack, it is clearly harder to maintain a high success rate using the publish attack, due to the permanent arrivals of new peers and the need to re-attack several peers periodically. While the node insertion attack yields constantly high rates, this is not true for the publish attack. Figure 12.2 plots the success rate of an attack on the keyword "Simpsons" over a period of 5 days. The attack works fairly well on average, at a success rate of roughly 80%, but the success rate periodically drops and remains low for a certain time period before it recovers again.

Overall, the success rate is much lower than in the case of a node insertion attack, although performing a publish attack is more expensive. Again, repeating the attack at other times using different keywords results in a similar pattern. The reason for this peculiar behavior is that the peers responsible for the targeted IDs that are online during the phase where the success rate is low refuse to accept our publish messages. In fact, these peers do not even reply to publish messages, even though they can be contacted, otherwise we could not receive any lookup results from them. As this behavior is not in accord with the protocol implemented in the real eMule client, we suspect that modified versions of the original clients cause this phenomenon. What clients are used is hard to determine as they do not directly provide this information. Thus, the use of modified clients appears to be another reason

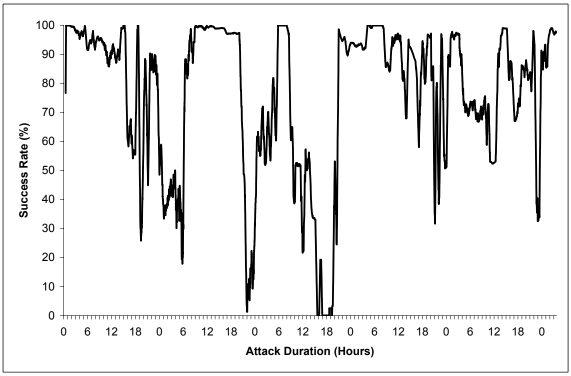


Figure 12.2: Percentage of faked replies received in a publish attack for the keyword "Simpsons" during a time period of 5 days. Sometimes, the success rate drops but then recovers again quickly.

why the node insertion attack is superior to the publish attack.

12.2.3 Eclipse Attack

Instead of poisoning the network to keep peers from obtaining certain information, we can also attack the requesting peers directly and keep them from sending requests into the Kad network. In the eclipse attack, the attacker takes over the targeted peer's routing table such that it is unable to communicate with any other peer in the Kad network except the attacker. As the attacker simulates the whole Kad network for that peer, it can manipulate the attacked peer in arbitrary ways, e.g., it can specify what results are returned for any lookup, or modify comments for any file. The peer's requests can also be directed back into the Kad network, but modified arbitrarily.

Typically, the contacts in the Kad routing table are not uniformly distributed over the whole ID space. Rather, most of the contacts are located around the peer's ID to maintain short lookup paths when searching for other peers in the Kad network (cf [161]). The attacker takes advantage of the fact that there are relatively few contacts in most parts of the ID space. Concretely, we inject faked peer entries into these parts of the routing table to achieve a dominating position. Subsequently, the faked peers are selected for almost all requests. If we set the IP address of all those faked entries to the address of our attacking peer, we receive most requests of the attacked peer and can process them as desired. We exploit the fact that the standard eMule client accepts multiple neighbors of the same IP address.

Our measurements showed that a peer running eMule for an extended time period has up to 900 contacts in its routing table. As the maximum number of contacts is 6,310, there is plenty of space in the routing table for the faked entries. To inject these faked entries the *Hello Request* message is used, which is normally utilized during connection set up to check whether known peers are still alive. As a side-effect, the sender of the message is added to the receiver's routing table. After enough entries are injected, the attacking peer has to process the requests from all those entries in order to keep them in the routing table of the attacked node.

We implemented the eclipse attack in a stand-alone application and ported all necessary parts from the source code of eMule. The application maintains a list that holds all faked entries sent to the attacked peer. This is necessary, because every new entry in the routing table is validated by sending a hello request. This request has to be answered with the same ID as we have chosen when injecting the entry. In order to differentiate between the entries, we assign a new port to every faked entry and maintain a data structure to store this information. The other part of our application processes the requests of the attacked peer. If it asks for new peers close to a specific ID, we reply with new faked peers that match this ID, or are very close to it, to guarantee the success of the attack. If the peer asks for stored information we deliver poisoned results, as in the two attacks discussed before.

Our experiments have revealed that the eclipse attack is effective, in particular if only certain keywords are targeted. In that case, the attacker only has to partially fill the routing table of the attacked peer, which renders the attack more efficient. The success rate virtually always reaches 100% within seconds and hence the attack works well, especially if it is focused on a single keyword; however, the attack is naturally limited to merely a single attacked peer. The other two attacks are clearly preferable if an attacker aims at hiding content from *all* peers.

12.3 Implications

The preceding section has presented three different attacks that can be used to keep peers from acquiring the requested information. Naturally, these exploits can also be combined in order to increase the chances of a success. However, the poisoning attacks cannot only be used for this purpose. Rather, they can serve an attacker as basic building blocks to pursue completely different aims.

We will now briefly illustrate another attack. The resources of the Kad network's peers and our attacks can be used to drive a *distributed denial of service attack* (DDoS) against any machine internal or external to the Kad network as follows: a node insertion attack is performed in order to occupy some popular keywords. Let μ be the machine (e.g., a server) to be attacked. We inform all requesters that μ contains the desired files. Consequently, all requests are directed to the attacked machine. Of course, the resulting load on μ is not larger than on the machine performing the node insertion. However, the advantage of this attack is that the attacking machine *remains hidden*; moreover, it is generally harder to counter a distributed DoS attack than a normal DoS attack as the request originate from different (and valid)

IP addresses. Also the publish attack can be used for the DDoS attack if we advertise wrong IP-bindings of keywords. This has the additional advantage that the attack induces more load on the victim than on the attacker, as the different Kad peers are directed to the victim directly. Note that DDoS attacks using a p2p system such as Kad are particularly harmful as the peers store information about sources for a long period of time, implying that such an attack could last several days with steadily changing peers involuntarily performing the attack.

As all the described attacks can be performed easily and have a large impact, it is mandatory to derive and implement counteractive measures. In order to overcome the node insertion attack it must be guaranteed that choosing specific IDs is infeasible. A straightforward approach, which is often described in literature, is to bind the ID directly to the peers' IP addresses, e.g., by hashing the IP address. However, there are several reasons why real-world p2p systems do not adhere to this simple rule. First, multiple peers may share the same IP address, for example, peers in a local area network behind a NAT router are typically addressed using the same public IP address. These peers would all have the same peer identifier. Second, IP addresses are often given out dynamically and the assignment of addresses may change. In case of an ID-IP binding, this implies that peers have to rebuild their routing tables when reconnecting to the network with a new IP. Additionally, all the credits gathered by uploading data would be lost irretrievably because the peer ID changed and hence the peer cannot be recognized by other peers anymore. It seems that some of these problems can be solved and the IP address can still be incorporated into the ID, e.g., by hashing the IP address and a randomly chosen bit string to solve the NAT problem, or by using a different, randomly chosen ID for the credit system, together with a public and private key pair to protect it against misuse.[3] Hashing the IP address and a user-generated bit string is preferable to including the port as this would require a static assignment of ports, and switching ports would also lead to a new ID. However, the crucial observation is that creating such a binding is not sufficient to avert the attack in general, as long as the ID includes a user-generated part. Assuming that a hash function such as SHA-1 is used, an attacker can try out millions of bit string in a short period of time in order to generate an ID that is closest to the targeted keyword even in a network containing more than a million peers. Thus, another form of peer authentication would be required which is hard to achieve without the use of a centralized verification service. As part of the strength of the network is its completely decentralized structure, relying on servers does not seem to be an acceptable solution.

A simple heuristic to render the Kad network more resilient to publish and eclipse attacks is to limit the amount of information a peer accepts from the same IP address, i.e., a peer does not allow that its entire contact list is filled by peers using the same IP address. This is also a critical solution as several peers behind a NAT may indeed have the same public IP address. What is

[3]In fact, Kad already uses public and private keys to authenticate peers whenever a new session starts.

more, an attacker with several IP addresses at its disposal can circumvent this security measure. We conclude that straightforward modifications may lead to an increased robustness, but a powerful attacker can nevertheless launch various effective attacks, and deriving strong disincentives is challenging. Furthermore, a crucial observation is that many of the discussed vulnerabilities do not only pertain to the Kad network, such attacks can be launched against any fully decentralized system that does not incorporate strong verification mechanisms.

12.4 Concluding Remarks

Our case study has provided evidence that the Kad network, which is currently the only widely deployed p2p network based on a DHT, can be attacked with a small amount of computing resources such that access to popular files is denied. It is clear that such attacks could significantly lower the throughput of the entire system as the sought-after files are no longer found, and that this imposed censorship would frustrate the users. Moreover, the possibility of leveraging the immense computational resources of the entire system to attack arbitrary machines constitutes a serious threat. We argue that the presented attacks can basically be launched in any peer-to-peer system that does not incorporate sound peer authentication mechanisms. While certain vulnerabilities can be mitigated to a certain extent, more research is needed on how to avert attacks on p2p networks, such as those presented in this chapter.

Chapter 13

Related Work

Papadimitriou [183] has argued that the Internet has surpassed the *von Neumann computer* as the most complex computational artifact of our time. In particular, he pointed out that the Internet has a *socio-economic* complexity whose understanding requires techniques from mathematical economics and game theory [180]. Since then, game theoretic approaches have become increasingly popular to study selfish behavior on all layers of distributed systems. Specifically, researchers have been keen to study the inherent loss of efficiency in a system caused by the participant's selfishness in networks. Consequently, the Price of Anarchy and its complexity have been investigated in various system settings, for example in *routing* [67, 206] (see also the book by Roughgarden [205]): Roughgarden has observed that the data packets which usually "rocket along the Internet at the speed of light" [29] can be significantly slowed down by selfish routers—a phenomenon which has been known also in civil engineering in the context of road planning. Besides being a useful toolkit to understand strategic interactions occurring in today's Internet, game theory itself offers exciting questions for mathematicians and computer scientists. For instance, while it is known that Nash equilibria always exist if the players' strategies are convex sets (such sets can for example be obtained by using probability distributions over strategies), in general, the complexity of finding a Nash equilibrium is believed to be one of the most important open questions on the boundary of the complexity class **P**, besides *factoring* [183]. Despite decades of effort, the computational complexity of computing a Nash equilibrium for a general-sum normal-form game remains unknown.

Economic literature has studied game theory for several decades, for various problems from *climate change* to questions related to *war and peace* [28]. Since the influential work of brilliant mathematicians like John von Neumann, Oskar Morgenstern or John Nash [172, 238, 239], a great number of results have been obtained in this area. In this endeavor, several generalizations have been proposed, taking into account many aspects of reality. A prime example is Harsanyi's historical work on *incomplete information games* [110] which opened the door to many new economic fields. Before this work, it has been thought that game theory is only applicable if the payoff matrix is completely known [71]. Also the rationality frontier has been weakened and there are suggestions that game theory need not even assume that players be utility maximizers [27]. Recently, *quantum game theory* [44] has shown that there exist interesting new solutions to classic problems like the prisoner's dilemma.

Adar and Huberman [5] noticed that selfish behavior is a reality also in peer-to-peer systems, and that there exists a large fraction of free riders in the file sharing network Gnutella. The problem of selfish behavior in peer-to-peer systems has been a hot topic in p2p research ever since, e.g. [113, 216], and many mechanisms to encourage cooperation have been proposed, for example in [91, 105, 209, 231, 237]. Perhaps the simplest fairness mechanism is to directly incorporate contribution monitoring into the client software. For instance, in the file-sharing system *Kazaa*, the client records the contribution of its user. However, such a solution can simply be bypassed by implementing a different client that hard-wires the contribution level to the maximum, as it was the case with *Kazaa Lite*. Inspired by real economies, some researchers have also proposed the introduction of some form of virtual money which is used for the transactions. However, these monetary or credit based approaches have a substantial overhead in terms of communication costs and infrastructure, and are inefficient [100, 241]. Often, these systems also require market regulating mechanisms [237] to cope with inflation or deflation—a complex issue. Additionally, monetary based systems may deter users from participating [177].

BitTorrent [66] has incorporated a fairness mechanism from the beginning. Although this mechanism has similarities to the well known *tit-for-tat scheme* [40], the mechanism employed in BitTorrent distinguishes itself from the classic tit-for-tat mechanism in many respects [118]. It has also been the subject of active research recently (e.g., [48, 99, 196]). Based on PlanetLab tests, [118] has argued that BitTorrent lacks appropriate rewards and punishments and therefore peers might be tempted to freeload. The authors further propose a tit-for-tat-oriented mechanism based on the iterated prisoner's dilemma [40] in order to deter peers from freeloading. However, in their work, a peer is already considered a free rider if it contributes considerably less than other peers. BitThief, on the other hand, aims at attaining fast downloads strictly without uploading any data. This is often desirable, since in many countries downloading certain media content is legal whereas uploading is not.

In [150], Liogkas et al. implement three selfish BitTorrent exploits and evaluate their effectiveness. They come to the conclusion that while peers can sometimes benefit slightly from being selfish, BitTorrent is fairly robust. Our case study extends [150] in that, rather than concentrating on individual attacks, we have implemented a client that combines several attacks (an open question in [150]). In contrast to the work presented in Chapter 8, the authors examine the effect of free riders on the *overall system* and argue that the quality of service is not severely affected by the presence of some peers that contribute only marginally. We have focused strictly on maximizing the download rate of a *single, selfish peer*, regardless of what effect this peer has on the system. In [16], the cooperation in *BitTorrent communities* has been studied. It has been shown that community-specific policies can boost cooperation. We have demonstrated that cheating is often easy in communities and selfish behavior is even more rewarding.

Shortly after the publication of BitThief, Piatek et al. [190] announced their *BitTyrant* client—a modification of the Azureus client implemented in Java. BitTyrant's strategy is to exploit the BitTorrent protocol in order to maximize download rates. In contrast to BitThief, BitTyrant does not free ride. For instance, BitTyrant uses a smart neighbor selection strategy and connects to those peers with the best reciprocation ratios. BitTyrant seeks to provide the minimal necessary contribution, and also increases the active set if this is beneficial to the download rate. The authors claim that their client provides a median 70% performance gain in certain environments. Finally, note that in a work by Shneidman, Parkes and Massoulié [217], a manual backtracing method is introduced which allows to discover potentially harmful protocol manipulations of protocols in an algorithmic way; with their backtracing graph, the faithfulness of Internet algorithms such as BitTorrent can be analyzed. So far, we have not made use of their approach.

The game-theoretic model of our locality game (cf Chapter 9) has been inspired by the paper by Fabrikant et al. [87] which studies the Internet's architecture as built by economic agents, e.g., by Internet providers or *autonomous systems*. Recent subsequent work on network creation in various settings includes [12, 21, 60, 69, 75, 84]. In contrast to all these works, our model takes into account many of the intrinsic properties of p2p systems. For instance, it captures the important *locality properties* of p2p systems, i.e., the desire to reduce the latencies (expressed as the stretch) experienced when performing look-up operations. Furthermore, the fact that a peer can decide to which other peers it wishes to store pointers and thus maintain links yields a scenario with *directed* links. Building structured systems that explicitly exploit locality properties has been a flourishing research area in networking and p2p computing (e.g. [2, 207, 243]). In early literature on distributed hash tables, the major measure of system quality has been the number of hops required for look-up operations. While this hop-distance is certainly of importance, it has been argued that the delay of communication (i.e., the stretch between pairs of peers) is a more relevant quality measure. Based on results achieved in [191], systems such as [2, 4, 207, 248] guarantee a provably bounded stretch with a limited number of links per peer. All of these systems are structured and peers are supposed to participate in a carefully predefined topology. Our work complements this line of research by analyzing topologies as they are created by *selfish peers*, which are interested only in optimizing their *individual* trade-off between locality and maintenance overhead.

Security and robustness of distributed systems against malicious or Byzantine faults are of prime importance and have been an active field of research for many years. For instance, in [58], a Byzantine-fault-tolerant distributed file system has been proposed (cf also the Microsoft's distributed FARSITE system [6]). Possibly the most well-known problem in this context has been that of reaching *consensus* among distributed parties. Possibility and impossibility results on the Byzantine consensus problem have been achieved in a variety of models and settings. Classic work in the synchronous and asynchronous case includes [78, 138, 218] and [94], respectively. In addition to the consensus problem, the distributed computing community has come up with results and solutions for a wide variety of other problems with Byzantine faults. Examples are *clock synchronization* [242], *broadcast* [134, 222], or *quorum systems* [159]. All of the above works assume that non-Byzantine players (or processes) are benevolent and attempt to reach a common goal. Finally, Byzantine behavior is subject to intensive research in cryptography. For instance, there is a large body of work in the area of *secure multi-party computation* [246]. In this context, two interesting papers by Halpern and Teague [109] and by Abraham et al. [3] need to be mentioned which consider self-interested players, e.g., in secret sharing problems. In [3], it is shown that Nash equilibria exist for secret sharing where no member of a coalition of size up to k nodes can do better even if the entire coalition defects, provided that players prefer to get the secret information than not to get it.

Chapter 10 strives for combining fault-tolerance research with game theory. In this respect, our work is related to the notions of *fault tolerant implementation* introduced by Eliaz [85] and of *BAR fault tolerance* introduced by Aiyer et al. [9]. In [85], implementation problems are investigated with k faulty players in the population, but neither their number nor their identity is known. A planner's objective then is to design an equilibrium where the non-faulty players act according to his rules. In [9], the authors describe an asynchronous state machine replication protocol which tolerates Byzantine, Altruistic, and Rational behavior. Interestingly, they find that the presence of Byzantine players can simplify the design of protocols if players are risk averse. In contrast to our work, rather than evaluating the degradation due to non-cooperative behavior, a concrete protocol for a *cooperative backup service* is provided.

To the best of our knowledge, the first paper to study equilibria with a malicious player is by Karakostas and Viglas [121]. They consider a routing application where a single malicious player uses his flow through the network in an effort to cause the maximum possible damage. In order to evaluate the impact of such malicious behavior, a coordination ratio is introduced which compares the social costs of the worst *Wardrop equilibrium* to the social costs of the best *minimax saddle-point.* A different basing point is used where the benevolent coordinator cannot influence the malicious player: in the "social optimum" only the costs of the players which can be coordinated are minimized, while the malicious player seeks to maximize costs.

There exists other work on game theoretic systems in which not every participating agent acts in a rational or selfish way. In the *Stackelberg theory* [204], for instance, the model consists of a set of selfish players, but a certain fraction of the entire population is *controlled by a global leader.* The leader's goal is to devise a strategy that induces an optimal or near optimal so-called Stackelberg equilibrium.

An interesting follow-up work by Babaioff, Kleinberg and Papadimitrou analyzes the so-called *Windfall of Malice* in the context of *non-atomic conges-*

tion games [41]. The authors investigate pure and mixed equilibria. Their players are less risk-averse, and a different solution concept is employed. Consider a game for three companies with three production strategies: *clean* production of a good, production of a good which *pollutes* the environment, and environmental *inspection* of the other companies. In their model, there is a unique equilibrium where all three companies choose a polluting strategy. Now assume a malicious company whose utility is the negative sum of the other two companies' payoffs. It can be shown that this malicious player selects the inspection strategy, which yields an equilibrium where the two remaining companies go for a clean production. Eventually, the social welfare is *increased*, and hence, a Windfall of Malice indeed exists. The author conjecture that there is no Windfall of Malice in their model in networks where the set of paths is a *matroid*, and also find that the absence of a Windfall of Malice may be related to the absence of some sort of *generalized Braess' paradox*. Finally, note that there also exists work on auctions with agents that derive utility from the disutility of others [55, 166]. Both papers derive symmetric Bayes Nash equilibria for spiteful agents in 1st-price and 2nd-price sealed bid auctions. The authors show that the revenue equivalence between second-price and first-price auctions breaks down with spiteful agents, with second-price outperforming first-price. Finally, a recent work by Roth [203] has studied the Price of Malice in linear congestion games, making weaker assumptions on the behavior of rational and malicious player, and using no-regret analysis.

Of course, coordination and collaboration problems involving malicious players also exist outside game theory. To just give one example, consider the *collaborative filtering* problem studied in [32] (cf also [31]), where there are Byzantine players among the n players participating in a reputation system like eBay. The goal of the honest players is to find a good object, and they can use a shared billboard to collaborate. The dilemma of an honest player is how to balance between the desire to reduce her cost by taking advantage of the reports posted by honest peers, and the fear of being exploited by adopting reports posted by malicious players.

There exist many virus propagation models in literature. While traditional epidemiological models characterize infection in terms of birth rate and death rate of the virus [43], more recently, models have been proposed for all kind of graphs, including Internet-like power-law graphs [185]. In particular, the game theoretic virus propagation model of Chapter 10 is based on [24]. The authors of [24] model the containment of the spread of viruses in general graphs. They characterize equilibria in selfish environments and also give an approximation algorithm for the centralized, non-selfish case.

In Chapter 11, the virus inoculation game has been studied on *social networks* which lay equal claims on the interest of social scientists, biologists, ethnologists, psychologists, linguists, and communication scientists. In these networks, nodes (individuals or organizations) are tied by a specific type of interdependency, e.g., friendship, disease transmission, economic trade, sexual relations, web links, and so on. In the last years, social network analysis has moved from being a suggestive metaphor to an analytical paradigm, and many different metrics have been explored on such graphs (e.g., *centrality* *indices* or *clustering coefficients*) [54]. The power of social network analysis stems from its new perspective where the attributes of individuals are less important than their relationships with other actors in the network. This approach turned out to be useful to explain many phenomena.

Computer scientists are also attracted by these networks. Many social networks exist today on the Internet, e.g., *Facebook* [88], *LinkedIn* [149], *MySpace* [168], *Orkut* [179], or *Xing* [244], to name but a few. The popular *small world experiment* [163] conducted by Stanley Milgram 1967 has gained attention by the algorithm community [131] and sparked research on topics such as decentralized search algorithms [132, 160], routing [98, 131, 148] and the identification of communities [95, 173]. Computer scientists have also proposed to use social networks to circumvent censorship, as millions of people today suffer from censorship (blockage of various websites ranging from YouTube, Flickr, Wikipedia or even Google have been reported) [221].

The dynamics of epidemic propagation of information or diseases has been studied from an algorithmic perspective as well [133, 142]. Knowledge on effects of the cascading behavior in social networks sheds light on phenomena as diverse as *word-of-mouth effects*, the diffusion of innovation, the emergence of bubbles in a financial market or the rise of a political candidate. It can also help to identify sets of influential nodes in networks where marketing is particularly efficient (*viral marketing*). For a good overview on economic aspects of social networks, we refer the reader to [51], which also compares random graph theory with game theoretic models for the formation of social networks.

Recently, several misuses of social networks by viruses have been reported: for instance, *email viruses*[1] have used address lists to propagate to the users' friends. Similar vulnerabilities have been exploited to spread worms on the *mobile phone network* [96] and on the Internet telephony tool *Skype*[2].

There is also other literature on game theory where players are influenced by their neighbors. In *graphical economics* [120, 127], an undirected graph is given where an edge between two players denotes that free trade is allowed between the two parties, where the absence of such an edge denotes an embargo or an other restricted form of direct trade. The payoff of a player is a function of the actions of the players in her neighborhood only. In contrast to our work, a different equilibrium concept is used and no social aspects are taken into consideration.

Chapter 12 has investigated the feasibility of malicious attacks on Kad. It is well-known that the immense computational resources of p2p networks can often be exploited by attackers, and there is already a large body of literature on the subject [57, 240].[3] Reasons to attack a p2p system can be manifold: besides the more or less passive "rational attacks" [174], a malicious attacker may, for example, strive to partition the system or to eclipse individual nodes. The *eclipse attack* [219] can be used by a set of malicious peers to position themselves around a given peer in the network

[1]E.g., the Outlook worm *Worm.ExploreZip*.

[2]See http://news.softpedia.com/news/Skype-Attacked-By-Fast-Spreading-Virus-52039.shtml.

[3]See also http://www.prolexic.com/news/20070514-alert.php/.

such that the peer's contact list consists only of the colluding peers. In a *Sybil attack* [81], a single entity creates multiple entities of itself in order to gain control over a certain fraction of the system. Such an attack can undermine redundancy mechanisms and is hard to counter in a completely decentralized environment. Furthermore, the resources of a p2p system can be used to attack *any* machine connected to the Internet regardless of whether it is part of the p2p network or not. A *denial of service attack* can be launched in various p2p systems, e.g., Gnutella [26], Overnet [170], and BitTorrent [74]. During this attack, information about the victim, i.e., the targeted machine in the attack, is spread in the system. The victim is falsely declared as an owner of popular content, causing other peers searching for this content to contact the victim repeatedly. In BitTorrent, tracker information can be faked which leads peers to believe that the victim is a tracker for the desired content [74]. In the Kad network, DoS attacks can be launched by means of a redirection attack where a queried peer, the attacker, will return a response containing the address of the victim [230]. The attacks presented in this thesis can also be used to launch a DoS attack. The work closest in spirit to the one presented in Chapter 12 is the study of *index poisoning attacks* in FastTrack and Overnet [146]. Their index poisoning attack is akin to our publish attack where bogus information is pushed aggressively to the nodes responsible for the desired keywords. However, while this attack is also quite successful, it is not as effective in the Kad network as it is in FastTrack and Overnet. We showed that a different, even simpler poisoning attack is feasible. Moreover, our study of attacks in the Kad network is not limited to content poisoning and index poisoning, but also considers the eclipse attack to prevent peers from accessing a specific file. It is also worth pointing out that, in comparison to Kad, it is generally easier to perform attacks on Overnet, as it, e.g., does not check whether the sender of a publish message provided its own IP address as the owner of the file, and no cryptography is used for authentication.

While we believe that there are methods to contain the potential damage caused by some of the attacks presented in Chapter 12 to some extent, it is known that certain attacks require a logically centralized entity [81]. There exists also an interesting theoretic work by Awerbuch and Scheideler on identifying and excluding large sets of colluding peers [38]. However, these results cannot be used to counter our attacks as we require only a very small number of attackers close to a given ID, which is not sufficient to raise suspicion. Recently, the same authors [37] have proposed a denial-of-service resistant DHT. Concretely, a DHT is described which is robust to *past insider attacks*, i.e., against an adversary who knows everything about the system up to some time point t_0 not known to the system. The problem of how to protect peers in the DHT from participating in an attack themselves is not considered. For a more thorough discussion of possible countermeasures against attacks in p2p networks, the reader is referred to the corresponding literature [240].

Finally, there are several studies of the *Kad network* itself. Stutzbach et al. [228] describe implementation details of Kad in eMule, and [227] presents crawling results on the behavior of Kad peers. Steiner et al. [223] crawled the Kad network during several weeks and found e.g. that different classes of participating peers exist inside the network. We have recently also conducted a measurement study with an emphasis on the user behavior in the server-based eDonkey network and in the server-less Kad. Our results are described in [154].

Chapter 14

Conclusion

Since the introduction of Napster (in June 1999) and Gnutella (in March 2000), the proliferation of peer-to-peer technology has enabled millions of individuals from all over the planet to self-organize and collaborate in the dissemination of various contents. However, already 6 months after Gnutella's public release—and long before the first copyright infringement lawsuits—two thirds of the Gnutella network consisted of free riders who did not upload anything at all. [175]

Non-cooperation is a threat to the p2p paradigm which relies on voluntary resource contributions. Part II of this thesis has aimed at gaining deeper insights into the effects of both selfish and malicious behavior. We have shown that free-riding in today's BitTorrent is possible despite the tit-for-tat inspired barter algorithms. Free-riding can be particularly attractive for users in countries where uploading is expensive or prohibited by law. Moreover, it can be appealing to certain "malicious" participants to harm the system's availability or performance, e.g., because copyrighted music and movies are distributed illegally.

Why do today's systems still not effectively solve the cooperation challenge? Certainly, one explanation is that finding good solutions is difficult: p2p networks are typically large, have a high turn-over rate and provide a certain user anonymity. Many defections cannot be detected and peers can often create multiple identities for free. How difficult it is to design cooperation mechanisms also depends on the application. In case of live streaming, peers can be punished more easily as already a small fraction of late packets can reduce the streaming quality significantly. Our streaming tool Pulsar employs a time-constrained version of the tit-for-tat mechanism [152]. The bootstrap problem is tackled by providing only a subset of blocks for free to a peer, and this subset depends on the peer's ID. For on-demand p2p streaming, the picture looks different: peers which started to watch the movie later in time do not have any data to contribute to earlier peers. The online storage tool Wuala applies a shared history approach [105]: due to coding, there are many transactions and hence repeated interactions between the same peers. Finally, in the grid computing project, we exploit the presence of a centralized entity to monitor the collaboration: our server assigns virtual credit points to peers depending on their CPU cycles. These points allow to reward hard-working clients in different ways; for instance, there is a project website listing the project's top overall contributors, or the "user of the day" [45]. Alternatively, the credit points could be used in a *lottery* [82] which offers the chance of monetary compensations for active participants. In order to prevent that peers obtain credits for free, a distributed checking algorithm [137] is applied which verifies whether the peer's results have been computed properly.

Another question is of course why the recording industry, e.g., the *Recording Industry Association of America* (RIAA)[1], does not make use of the vulnerabilities presented in Chapter 12 in order to remove copyrighted material. One explanation is that these vulnerabilities may not be known yet. However, it is likely that a different strategy is pursued: instead of actively harming the infrastructure, a passive approach is taken in the sense that contravention is monitored and logged.[2]

The goal of designing better cooperation algorithms should be a driving force when setting up future p2p research agendas. Currently, game theory is our power house to improve our scientific understanding of the economic aspects of peer-to-peer computing. Game theory allows us to identify weaknesses in protocols and to give formal proofs on the incentive-compatibility of a proposed solution. Of course, the obtained conclusions cannot be strong if the underlying assumptions are unrealistic. This thesis has argued that providing the right incentives for selfish players alone is not sufficient to solve the non-cooperation problem, as there might be malicious players benefitting from a reduced social welfare in a system. A system should try to prevent such players from joining the network or to apply other means *external to the system* to ensure a good performance.

Peer-to-peer systems probably include many more types of players besides altruistic, selfish or malicious ones. It is impossible to assess all these players' cost or utility functions, and such behavior is often outside of typical game-theoretic models. People also react differently to the provided cooperation incentives. While some users can be motivated by the "geek chic" [82] associated with high contribution levels published on web sites others may

[1]See http://www.riaa.org/.

[2]The RIAA claims that p2p software results in a reduction of profits of 4.2 billion USD per year for the music industry worldwide [201]. Many lawsuits against users have been filed in the last years, which has been described by the New York Times as a campaign to raise the users' awareness that not everything is free in the Internet. However, note that the logging of IP addresses (and subsequently trying to find the corresponding users' real addresses or phone numbers) can be problematic, as it threatens rights of privacy (e.g., see the Swiss *Bundesgesetz über den Datenschutz* (DSG)).

Also observe that recently, there have indeed been reports on attacks by specialized anti-p2p companies working on behalf of the RIAA and specific record labels [76]. The music and film industry have started hiring companies to impede specific "assets" from being distributed, e.g., in BitTorrent.

Today, many judical questions remain open, and it is believed that it will take the laws of society some time to catch up with the technological shift introduced by p2p software [197].

demand monetary compensations. Moreover, peer-to-peer games are often dynamic games in the sense that the same players interact repeatedly in complex ways, where the individual players' utility functions can also evolve over time. It is possible that a player chooses a strategy that might seem irrational in the short term but can be beneficial in the long term. Experiments have also shown that people often do not play the Nash equilibrium strategy (cf also the paradox of the *traveler's dilemma*). Moreover, many studies today, including the one presented in this thesis, assume that nodes have a global knowledge of the network's state. In a distributed setting, it takes time to gather such information, and it would be interesting to adapt the equilibrium concepts for the case where the nodes' horizon is bounded. In addition, computation and communication power of players is often limited, and they may fail to pick the strategy maximizing the expected utility (bounded-rational players). Finally, it is well-known (e.g., cf [72, 90, 165, 216]) that also the network topology can have an influence on the design of a (distributed) algorithmic incentive mechanism: in a message-passing environment, during an auction on a large p2p network for example, it can be rational for nodes to drop messages instead of forwarding them to their neighbors.

Ignoring some of these aspects which are important in practice can distort the results, or even render them useless in practice. One difficulty with game theory is that it is hard to obtain impossibility results or lower bounds, as the model's point of view is not normative but rather analytical. For comparison, consider a connectivity model for wireless sensor networks. If we can prove the correctness and performance of an algorithm for a unit disk graph, this algorithm might fail after deployment. However, when we can give such a formal proof for general connectivity graphs, then we can be pretty sure that connectivity will not be a problem for this algorithm in practice. However, obtaining similar guarantees for selfish environments is difficult: assuming the most general utility functions is likely to render an analysis cumbersome and will not provide many hard results. Observe that such a system would essentially consist of an entirely Byzantine population.

Much game-theoretic research today is conducted on specific network layers. However, it is not clear whether all these protocols on the different layers can be seamlessly combined, and it would be interesting to further investigate the feasibility of an entire incentive-compatible protocol stack, not only for p2p systems, but also for other distributed systems like wireless sensor networks.

In a broader context, cooperation and collaboration are of course very fundamental subjects of research in many distributed systems or societies. An interesting example is the *Wikipedia* project which has managed to motivate thousands of volunteers around the world to contribute articles to the online encyclopedia. Also here, despite the great success, many challenges remain. For instance: how can the quality of Wikipedia articles be guaranteed or even improved in future without sacrificing the principles of collaboration and scalability, i.e., without the need for supervisors? We believe that lessons learnt in such alternative environments could also be useful in the context of peer-to-peer computing. However, only future can tell whether algorithmic solutions to these interdisciplinary questions exist.

Bibliography

[1] I. Abraham, B. Awerbuch, Y. Azar, Y. Bartal, D. Malkhi, and E. Pavlov. A Generic Scheme for Building Overlay Networks in Adversarial Scenarios. In *Proc. 17th Int. Symposium on Parallel and Distributed Processing (IPDPS)*, 2003.

[2] I. Abraham, A. Badola, D. Bickson, D. Malkhi, S. Maloo, and S. Ron. Practical Locality-Awareness for Large Scale Information Sharing. In *Proc. 4th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2005.

[3] I. Abraham, D. Dolev, R. Gonen, and J. Halpern. Distributed Computing Meets Game Theory: Robust Mechanisms for Rational Secret Sharing and Multiparty Computation. In *Proc. 25th Annual Symposium on Principles of Distributed Computing (PODC), Denver, Colorado, USA*, 2006.

[4] I. Abraham, D. Malkhi, and O. Dobzinski. LAND: Stretch $(1+e)$ Locality Aware Networks for DHTs. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 550–559, 2004.

[5] E. Adar and B. A. Huberman. Free Riding on Gnutella. *First Monday*, 5(10), 2000.

[6] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer. FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment. In *Proc. 5th Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[7] V. Aggarwal, A. Feldmann, and C. Scheideler. Can ISPs and P2P Users Cooperate for Improved Performance? *ACM Computer Communication Review*, 2007.

[8] W. Aiello, E. Kushilevitz, R. Ostrovsky, and A. Rosén. Adaptive Packet Routing for Bursty Adversarial Traffic. In *Proc. 13th Annual ACM Symposium on Theory of Computing (STOC)*, pages 359–368, New York, NY, USA, 1998.

[9] A. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. BAR Fault Tolerance for Cooperative Services. In *Proc. 20th ACM Symposium on Operating Systems Principles (SOSP)*, pages 45–58, 2005.

[10] A. Akella, S. Seshan, and A. Shaikh. An Empirical Evaluation of Wide-Area Internet Bottlenecks. In *Proc. ACM Internet Measurement Conference (IMC)*, 2003.

[11] S. B. Akers and B. Krishnamurthy. A Group-Theoretic Model for Symmetric Interconnection Networks. *IEEE Transactions on Computing*, 38(4):555–566, 1989.

[12] S. Albers, S. Eilts, E. Even-Dar, Y. Mansour, and L. Roditty. On Nash Equilibria for a Network Creation Game. In *Proc. 17th ACM Symposium on Discrete Algorithms (SODA)*, 2006.

[13] K. Albrecht, R. Arnold, M. Gähwiler, and R. Wattenhofer. Aggregating Information in Peer-to-Peer Systems for Improved Join and Leave. In *Proc. 4th IEEE Int. Conference on Peer-to-Peer Computing (P2P)*, 2004.

[14] D. Aldous. Ultimate Instability of Exponential Back-off Protocol for Acknowledgement Based Transmission Control of Random Access Communication Channels. *IEEE Transactions on Information Theory*, 1987.

[15] D. P. Anderson. BOINC: A System for Public-Resource Computing and Storage. In *Proc. 5th IEEE/ACM Int. Workshop on Grid Computing (GRID)*, pages 4–10, 2004.

[16] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu. Influences on Cooperation in BitTorrent Communities. In *Proc. 3rd ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, 2005.

[17] M. Andrews, B. Awerbuch, A. Fernández, T. Leighton, Z. Liu, and J. Kleinberg. Universal-Stability Results and Performance Bounds for Greedy Contention-Resolution Protocols. *Jounal ACM*, 48(1):39–69, 2001.

[18] Andy Oram (Ed.). *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.

[19] D. Angluin, J. Aspnes, J. Chen, Y. Wu, and Y. Yin. Fast Construction of Overlay Networks. In *Proc. 17th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 145–154, 2005.

[20] F. Annexstein, M. Baumslag, and A. L. Rosenberg. Group Action Graphs and Parallel Architectures. *SIAM Journal Comput.*, 19(3):544–569, 1990.

[21] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The Price of Stability for Network Design with Fair Cost Allocation. In *Proc. 45th Symposium on Foundations of Computer Science (FOCS)*, pages 295–304, 2004.

[22] S. Arora and B. Brinkman. A Randomized Online Algorithm for Bandwidth Utilization. In *Proc. 13th Annual ACM Symposium on Discrete Algorithms (SODA)*, pages 535–539, Philadelphia, PA, USA, 2002.

[23] S. Ashby, G. Eulisse, S. Schmid, and L. Tuura. Parallel Compilation of CMS Software. In *Proc. Computing in High Energy and Nuclear Physics Conference (CHEP)*, 2004.

[24] J. Aspnes, K. Chang, and A. Yampolskiy. Inoculation Strategies for Victims of Viruses and the Sum-of-Squares Partition Problem. In *Proc. 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 43–52, 2005.

[25] J. Aspnes and G. Shah. Skip Graphs. In *Proc. 14th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 384–393, 2003.

[26] E. Athanasopoulos, K. G. Anagnostakis, and E. P. Markatos. Misusing Unstructured P2P Systems to Perform DoS Attacks: The Network That Never Forgets. In *Proc. 4th Int. Conference on Applied Cryptography and Network Security (ACNS)*, 2006.

[27] R. J. Aumann. Correlated Equilibrium as an Expression of Bayesian Rationality. *Econometrica*, 55:1–18, 1987.

[28] R. J. Aumann. War and Peace. *Nobel Prize Lecture, Discussion Paper No 428*, 2006.

[29] I. Austen. Like a Swerving Commuter, a Selfish Router Slows Traffic. *The New York Times*, 2003.

[30] B. Awerbuch, S. Kutten, Y. Mansour, B. Patt-Shamir, and G. Varghese. Time Optimal Self-Stabilizing Synchronizers Using Phase Clocks. *IEEE Trans. on Dependable Systems*, 2007.

[31] B. Awerbuch, B. Patt-Shamir, D. Peleg, and M. Tuttle. Collaboration of Untrusting Peers with Changing Interests. In *Proc. ACM Conference on Electronic Commerce (EC)*, pages 112–119, 2004.

[32] B. Awerbuch, B. Patt-Shamir, D. Peleg, and M. Tuttle. Adaptive Collaboration in Peer-to-Peer Systems. In *Proc. 25th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 71–80, 2005.

[33] B. Awerbuch, B. Patt-Shamir, and G. Varghese. Self-Stabilization By Local Checking and Correction. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, 1991.

[34] B. Awerbuch and C. Scheideler. The Hyperring: A Low-Congestion Deterministic Data Structure for Distributed Environments. In *Proc. 15th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 318–327, 2004.

[35] B. Awerbuch and C. Scheideler. Robust Random Number Generation for Peer-to-Peer Systems. In *Proc. 10th International Conference on Principles of Distributed Systems (OPODIS)*, 2006.

[36] B. Awerbuch and C. Scheideler. Towards a Scalable and Robust DHT. In *Proc. 18th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 2006.

[37] B. Awerbuch and C. Scheideler. A Denial-of-Service Resistant DHT. In *Proc. 21st Ann. Conference on Distributed Computing (DISC)*, pages 33–47, 2007.

[38] B. Awerbuch and C. Scheideler. Towards Scalable and Robust Overlay Networks. In *Proc. 6th Int. Workshop on Peer-To-Peer Systems (IPTPS)*, 2007.

[39] B. Awerbuch and M. Sipser. Dynamic Networks Are as Fast as Static Networks. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, 1988.

[40] R. Axelrod. The Evolution of Cooperation. *Science*, 211(4489):1390-6, 1981.

[41] M. Babaioff, R. Kleinberg, and C. Papadimitriou. Congestion Games with Malicious Players. In *Proc. ACM Conference on Electronic Commerce (EC), San Diego, CA, USA*, 2007.

[42] A. Bagchi, A. Bhargava, A. Chaudhary, D. Eppstein, and C. Scheideler. The Effect of Faults on Network Expansion. In *Proc. 16th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 2004.

[43] N. Bailey. The Mathematical Theory of Infectious Diseases and Its Applications. *Hafner Press*, 1975.

[44] S. C. Benjamin and P. M. Hayden. Multiplayer Quantum Games. *Phys. Rev.*, 2001.

[45] Berkeley Open Infrastructure for Network Computing. BOINC Combined Statistics. In *http://boinc.netsoft-online.com/*, 2006.

[46] K. Berman. Vulnerability of Scheduled Networks and a Generalization of Menger's Theorem. *Networks*, 28:125–134, 1996.

[47] R. Bhagwan, S. Savage, and G. Voelker. Understanding Availability. In *Proc. 2nd Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[48] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. Analyzing and Improving a BitTorrent Network's Performance Mechanisms. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*, pages 36–46, 2006.

[49] A. Bhargava, K. Kothapalli, C. Riley, C. Scheideler, and M. Thober. Pagoda: A Dynamic Overlay Network for Routing, Data Management, and Multicasting. In *Proc. 16th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 170–179, 2004.

[50] BitTorrent. Website. http://www.bittorrent.com/.

[51] R. Blundell, W. Newey, and Torsten Persson (Eds). Advances in Economics and Econometrics (Chapter 1: The Economics of Social Networks). 2006.

[52] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[53] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial Queuing Theory. In *Proc. 28th Annual Symposium on Foundations of Computer Science (STOC)*, 1996.

[54] U. Brandes and Thomas Erlebach (Eds). *Network Analysis.* Springer, 2005.

[55] F. Brandt, T. Sandholm, and Y. Shoham. Spiteful Bidding in Sealed-bid Auctions. In *Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[56] Caleido. Website. http://www.caleido.com/.

[57] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure Routing for Structured Peer-to-Peer Overlay Networks. In *Proc. 5th Symposium on Operating Systems Design and Implementation (OSDI)*, pages 299–314, 2002.

[58] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance. In *Proc. 3rd Symposium on Operating Systems Design and Implementation (OSDI)*, pages 173–186, 1999.

[59] H. T.-H. Chan, A. Gupta, B. M. Maggs, and S. Zhou. On Hierarchical Routing in Doubling Metrics. In *Proc. 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 762–771, 2005.

[60] H.-L. Chen and T. Roughgarden. Network Design with Weighted Players. In *Proc. 18th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 29–38, 2006.

[61] Christoph Renner (Advisors: Michael Kuhn and Stefan Schmid). BOINC and Distributed Checking. In *Semester Thesis*, 2007.

[62] Christoph Schwank (Advisors: Michael Kuhn and Stefan Schmid). BOINC and Distributed Checking. In *Master Thesis*, 2007.

[63] B.-G. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. H. Papadimitriou, and J. Kubiatowicz. Selfish Caching in Distributed Systems: A Game-theoretic Analysis. In *Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 21–30, 2004.

[64] D. Clark. Face-to-Face with Peer-to-Peer Networking. *Computer*, pages 18–21, January 2001.

[65] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley. Protecting Free Expression Online with Freenet. *IEEE Internet Computing*, 2002.

[66] B. Cohen. Incentives Build Robustness in BitTorrent. In *Proc. 1st Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, 2003.

[67] R. Cole, Y. Dodis, and T. Roughgarden. How Much Can Taxes Help Selfish Routing? *Journal Comput. Syst. Sci.*, 72(3):444–467, 2006.

[68] S. A. Cook. The Complexity of Theorem-Proving Procedures. In *Proc. 3rd ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 1971.

[69] J. Corbo and D. C. Parkes. The Price of Selfish Behavior in Bilateral Network Formation. In *Proc. 24th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 99–107, 2005.

[70] G. Cybenko. Dynamic Load Balancing for Distributed Memory Multiprocessors. *Journal on Parallel Distributed Computing*, 7:279–301, 1989.

[71] P. Dasgupta, D. Gale, O. Hart, and E. Maskin. Economic Analysis of Markets and Games: Essays in Honor of Frank Hahn. *MIT Press*, pages 214–227, 1992.

[72] R. K. Dash, N. R. Jennings, and D. C. Parkes. Computational-Mechanism Design: A Call to Arms. *IEEE Intelligent Systems*, 18(6):40–47, 2003.

[73] A. Datta and K. Aberer. Internet-Scale Storage Systems Under Churn. In *Proc. 6th IEEE Int. Conference on Peer-to-Peer Computing (P2P)*, 2006.

[74] K. E. Defrawy, M. Gjoka, and A. Markopoulou. BotTorrent: Misusing BitTorrent to Launch DDoS Attacks. In *Proc. 3rd Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2007.

[75] E. D. Demaine, M. T. Hajiaghayi, H. Mahini, and M. Zadimoghaddam. On the Topologies Formed by Selfish Peers. In *Proc. 26th Annual Symposium on Principles of Distributed Computing (PODC)*, 2007.

[76] P. Dhungel, D. Wu, B. Schonhorst, and K. W. Ross. A Measurement Study of Attacks on BitTorrent Leechers. In *Proc. 7th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2008.

[77] E. W. Dijkstra. Self-stabilizing Systems in Spite of Distributed Control. *Commun. ACM*, 17(11), 1974.

[78] D. Dolev. The Byzantine Generals Stike Again. *Journal of Algorithms*, 3(1):14–30, 1982.

[79] S. Dolev. *Self-Stabilization.* MIT Press, 2000.

[80] Dominik Grolimund and Luzius Meisser (Advisor: Stefan Schmid). Kangoo/Wuala (see http://www.caleido.com/kangoo/). In *Labs and Theses*, 2007.

[81] J. R. Douceur. The Sybil Attack. In *Proc. 1st Int. Workshop on Peer-to-Peer Systems (IPTPS)*, pages 251–260, 2002.

[82] J. R. Douceur and T. Moscibroda. Lottery Trees: Motivational Deployment of Networked Systems. In *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2007.

[83] H. Dweighter (a.k.a. J. E. Goodman). *American Mathematical Monthly*, 82, 1975.

[84] S. Eidenbenz, V. Kumar, and S. Zust. Equilibria in Topology Control Games for Ad Hoc Networks. In *Proc. ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, 2003.

[85] K. Eliaz. Fault Tolerant Implementation. *Review of Economic Studies*, 69:589–610, 2002.

[86] Emule Project. Website. http://www.emule-project.net/.

[87] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a Network Creation Game. In *Proc. 22nd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 347–351, 2003.

[88] Facebook. Website. http://www.facebook.com/.

[89] J. Feigenbaum, C. H. Papadimitriou, and S. Shenker. Sharing the Cost of Multicast Transmissions. *Journal of Computer and System Sciences*, 63(1):21–41, 2001.

[90] J. Feigenbaum and S. Shenker. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. In *Proc. 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM-POMC)*, pages 1–13, 2002.

[91] M. Feldman and J. Chuang. Overcoming Free-Riding Behavior in Peer-to-Peer Systems. *ACM Sigecom Exchanges*, 6, 2005.

[92] A. Fiat and J. Saia. Censorship Resistant Peer-to-Peer Content Addressable Networks. In *Proc. 13th Symposium on Discrete Algorithms (SODA)*, 2002.

[93] A. Fiat, J. Saia, and M. Young. Making Chord Robust to Byzantine Attacks. In *Proc. European Symposium on Algorithms (ESA)*, 2005.

[94] M. Fischer, N. Lynch, and M. S. Paterson. Impossibility of Distributed Consensus with one Faulty Processor. *Journal ACM*, 32(2):374–382, 1985.

[95] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee. Self-Organization and Identification of Web Communities. *Computer*, 35(3):66–71, 2002.

[96] C. Fleizach, M. Liljenstam, P. Johansson, G. M. Voelker, and A. Mehes. Can You Infect Me Now? Malware Propagation in Mobile Phone Networks. In *Proc. 2007 ACM Workshop on Recurring Malcode (WORM)*, pages 61–68, 2007.

[97] R. Flury and R. Wattenhofer. MLS: An Efficient Location Service for Mobile Ad Hoc Networks. In *Proc. 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2006.

[98] P. Fraigniaud, C. Gavoille, and C. Paul. Eclecticism Shrinks Even Small Worlds. In *Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 169–178, 2004.

[99] P. Ganesan and M. Seshadri. On Cooperative Content Distribution and the Price of Barter. In *Proc. 25th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 81–90, 2005.

[100] F. D. Garcia and J.-H. Hoepman. Off-Line Karma: A Decentralized Currency for Peer-to-Peer and Grid Applications. In *Proc. 3rd Applied Cryptography and Network Security (ACNS)*.

[101] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, January 1979.

[102] Gnutella. Website. http://www.gnutella.com/.

[103] P. B. Godfrey, S. Shenker, and I. Stoica. Minimizing Churn in Distributed Systems. In *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2006.

[104] D. Grolimund, L. Meisser, S. Schmid, and R. Wattenhofer. Cryptree: A Folder Tree Structure for Cryptographic File Systems. In *Proc. 25th IEEE Symposium on Reliable Distributed Systems (SRDS)*, 2006.

[105] D. Grolimund, L. Meisser, S. Schmid, and R. Wattenhofer. Havelaar: A Robust and Efficient Reputation System for Active Peer-to-Peer Systems. In *Proc. 1st Workshop on the Economics of Networked Systems (NetEcon)*, June 2006.

[106] S. Guha, N. Daswani, and R. Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *Proc. 5th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2006.

[107] K. P. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall. Improving the Reliability of Internet Paths with One-hop Source Routing. In *Proc. Symposium on Operating Systems Design & Implementation (OSDI)*, 2004.

[108] A. Haeberlen, A. Mislove, A. Post, and P. Druschel. Fallacies in Evaluating Decentralized Systems. In *Proc. 5th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2006.

[109] J. Halpern and V. Teague. Rational Secret Sharing and Multiparty Computation. In *Proc. 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 623–632, 2004.

[110] J. C. Harsanyi. Games of Incomplete Information Played by Bayesian Players. *Management Science*, 14, 1967.

[111] N. Harvey and J. Munro. Deterministic SkipNet. *Inf. Process. Lett.*, 90(4):205–208, 2004.

[112] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. SkipNet: A Scalable Overlay Network with Practical Locality Properties. In *Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS)*, 2003.

[113] D. Hughes, G. Coulson, and J. Walkerdine. Free Riding on Gnutella Revisited: The Bell Tolls? *IEEE Distributed Systems Online*, 6(6), 2005.

[114] Jan Kostka (Advisors: Thomas Moscibroda and Stefan Schmid). Byzantine Caching Game. In *Semester Thesis*, 2006.

[115] Jean-Luc Geering (Advisors: Thomas Locher and Stefan Schmid). Towards Peer-to-Peer Games. In *Master Thesis*, 2007.

[116] Jie Wu (Ed.). *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks.* Auerbach Publications, 2006.

[117] Joost. Website. http://www.joost.com/.

[118] S. Jun and M. Ahamad. Incentives in BitTorrent Induce Free Riding. In *Proc. 3rd ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, 2005.

[119] Jun Li (Advisors: Thomas Locher and Stefan Schmid). Implementation of eQuus. In *Lab*, 2007.

[120] S. M. Kakade, M. Kearns, and L. E. Ortiz. Graphical Economics. In *Proc. 17th Annual Conference on Learning Theory (COLT)*, 2004.

[121] G. Karakostas and A. Viglas. Equilibria for Networks with Malicious Users. *Mathematical Programming A*, 110(3):591–613.

[122] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. In *Proc. 29th ACM Symposium on Theory of Computing (STOC)*, pages 654–663, 1997.

[123] D. R. Karger and M. Ruhl. Finding Nearest Neighbors in Growth-Restricted Metrics. In *Proc. 34th ACM Symposium on Theory of Computing (STOC)*, pages 741–750, 2002.

[124] A. Karlin, C. Kenyon, and D. Randall. Dynamic TCP Acknowledgement and Other Stories about $e/(e-1)$. In *Proc. 41st Annual Symposium on Foundations of Computer Science (STOC)*, 2001.

[125] R. M. Karp. Reducibility Among Combinatorial Problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[126] R. M. Karp, E. Koutsoupias, C. H. Papadimitriou, and S. Shenker. Optimization Problems in Congestion Control. In *Proc. Symposium on Foundations of Computer Science (FOCS)*, pages 66–74, 2000.

[127] M. Kearns, M. Littman, and S. Singh. Graphical Models for Game Theory. In *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 253–260, 2001.

[128] F. Kelly. Mathematical Modelling of the Internet. *Mathematics Unlimited – 2001 and Beyond (Springer)*, 2001.

[129] D. Kempe, A. Dobra, and J. Gehrke. Gossip-Based Computation of Aggregate Information. In *Proc. 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003.

[130] D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and Interference Problems for Temporal Networks. In *Proc. 32nd ACM Symposium on Theory of Computing (STOC)*, 2000.

[131] J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *Proc. 32nd ACM Symposium on Theory of Computing (STOC)*, 2000.

[132] J. Kleinberg. Complex Networks and Decentralized Search Algorithms. In *Proc. International Congress of Mathematicians (ICM)*, 2006.

[133] J. Kleinberg. *Algorithmic Game Theory. Chapter 24 : Cascading Behavior in Networks: Algorithmic and Economic Issues.* N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani (Eds), Cambridge University Press, 2007.

[134] C.-Y. Koo. Broadcast in Radio Networks Tolerating Byzantine Adversial Behavior. In *Proc. 23rd ACM Symposium on the Principles of Distributed Computing (PODC)*, pages 275–282, 2004.

[135] K. Kothapalli and C. Scheideler. Supervised Peer-to-Peer Systems. In *Proc. International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)*, pages 188–193, 2005.

[136] F. Kuhn, T. Locher, and R. Wattenhofer. Tight Bounds for Distributed Selection. In *Proc. 19th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2007.

[137] M. Kuhn, S. Schmid, and R. Wattenhofer. Distributed Asymmetric Verification in Computational Grids. In *Proc. 22nd IEEE Int. Parallel and Distributed Processing Symposium (IPDPS)*, 2008.

[138] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[139] J.-Y. Le Boudec and P. Thiran. *Network Calculus.* Springer LNCS 2050 Tutorial, 2001.

[140] F. Leighton, B. Maggs, and S. Rao. Universal Packet Routing Algorithms. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 256–269, 1988.

[141] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann, 1991.

[142] J. Leskovec, L. A. Adamic, and B. A. Huberman. The Dynamics of Viral Marketing. *ACM Transactions on the Web*, 2007.

[143] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph Evolution: Densification and Shrinking Diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(2), 2007.

[144] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil. A Performance vs Cost Framework for Evaluating DHT Design Tradeoffs under Churn. In *Proc. 24th Annual IEEE Conference on Computer Communications (INFOCOM)*, 2005.

[145] X. Li, J. Misra, and C. G. Plaxton. Active and Concurrent Topology Maintenance. In *Proc. 18th Ann. Conference on Distributed Computing (DISC)*, 2004.

[146] J. Liang, N. Naoumov, and K. W. Ross. The Index Poisoning Attack in P2P File Sharing Systems. In *Proc. 25th Annual IEEE Conference on Computer Communications (INFOCOM)*, 2006.

[147] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the Evolution of Peer-to-Peer Systems. In *Proc. 21st Annual Symposium on Principles of Distributed Computing (PODC)*, pages 233–242, 2002.

[148] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic Routing in Social Networks. Number 33, pages 11623–11628. National Acad Sciences, 2005.

[149] LinkedIn. Website. http://www.linkedin.com/.

[150] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang. Exploiting BitTorrent For Fun (But Not Profit). In *Proc. 5th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2006.

[151] T. Locher, R. Meier, S. Schmid, and R. Wattenhofer. Push-to-Pull Peer-to-Peer Live Streaming. In *21st Int. Symposium on Distributed Computing (DISC)*, 2007.

[152] T. Locher, R. Meier, S. Schmid, and R. Wattenhofer. Fairness in Peer-to-Peer Live Streaming. In *Under submission.*, 2008.

[153] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free Riding in BitTorrent is Cheap. In *Proc. 5th Workshop on Hot Topics in Networks (HotNets)*, 2006.

[154] T. Locher, D. Mysicka, S. Schmid, and R. Wattenhofer. Server vs DHT: A Peer Activity Study in eDonkey & Kad. In *Under submission*, 2008.

[155] T. Locher, S. Schmid, and R. Wattenhofer. eQuus: A Provably Robust and Locality-Aware Peer-to-Peer System. In *Proc. 6th IEEE Int. Conference on Peer-to-Peer Computing (P2P)*, 2006.

[156] T. Locher, S. Schmid, and R. Wattenhofer. Rescuing Tit-for-Tat with Source Coding. In *Proc. 7th IEEE Int. Conference on Peer-to-Peer Computing (P2P)*, September 2007.

[157] Z. Lotker, B. Patt-Shamir, and A. Rosén. New Stability Results for Adversarial Queuing. *SIAM J. Comput.*, 33(2):286–303, 2004.

[158] P. Mahlmann and C. Schindelhauer. *Peer-to-Peer Netzwerke.* Springer, 2007.

[159] D. Malkhi and M. Reiter. Byzantine Quorum Systems. *Journal of Distributed Computing*, 11(4):203–213, 1998.

[160] G. S. Manku, M. Naor, and U. Wieder. Know thy Neighbor's Neighbor: the Power of Lookahead in Randomized P2P Networks. In *Proc. 36th ACM Symposium on Theory of Computing (STOC)*, 2004.

[161] P. Maymounkov and D. Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Proc. 1st Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[162] M. Mense and C. Scheideler. SPREAD: An Adaptive Scheme for Redundant and Fair Storage in Dynamic Heterogeneous Storage Systems. In *Proc. 19th Annual ACM-SIAM Symposium on Discrete Algorithm (SODA)*, 2008.

[163] S. Milgram. The Small World Problem. *Psychology Today*, pages 60–67, 1967.

[164] B. Mitra, S. Ghose, and N. Ganguly. Effect of Dynamicity on Peer-to-Peer Networks. In *Proc. 14th International Conference on High Performance Computing (HiPC)*, 2007.

[165] D. Monderer and M. Tennenholtz. Distributed Games: From Mechanisms to Protocols. In *Proc. 16th National Conference on Artificial Intelligence (AAAI)*, pages 32–37, 1999.

[166] J. Morgan, K. Steiglitz, and G. Reis. The Spite Motive and Equilibrium Behavior in Auctions. *Contricutions to Economic Analysis & Policy*, 2(1), 2003.

[167] D. Mosk-Aoyama and D. Shah. Computing Separable Functions via Gossip. In *Proc. 25th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 113–122, 2006.

[168] MySpace. Website. http://www.myspace.com/.

[169] M. Naor and U. Wieder. Novel Architectures for P2P Applications: the Continuous-Discrete Approach. In *Proc. 15th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 50–59, 2003.

[170] N. Naoumov and K. Ross. Exploiting P2P systems for DDoS attacks. In *Proc. 1st Int. Conference on Scalable Information Systems (INFOSCALE)*, 2006.

[171] Napster. Website. http://www.napster.com/.

[172] J. Nash. Equilibrium Points in n-Person Games. In *Proc. National Academy of the USA*, pages 48–49, 1950.

[173] M. E. J. Newman and M. Girvan. Finding and Evaluating Community Structure in Networks. *Physical Review E*, 69, 2004.

[174] S. J. Nielson, S. Crosby, and D. S. Wallach. A Taxonomy of Rational Attacks. In *Proc. 4th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, pages 36–46, 2005.

[175] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[176] R. O'Dell and R. Wattenhofer. Information Dissemination in Highly Dynamic Graphs. In *Proc. 3rd ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, 2005.

[177] A. M. Odlyzko. The Case Against Micropayments. In *Financial Cryptography*, pages 77–83, 2003.

[178] M. Onus, A. W. Richa, and C. Scheideler. Linearization: Locally Self-Stabilizing Sorting in Graphs. In *Proc. Workshop on Algorithmic Engineering & Experiments (ALENEX)*, 2007.

[179] Orkut. Website. http://www.okkut.com/.

[180] M. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, 2000.

[181] G. Pandurangan, P. Raghavan, and E. Upfal. Building Low-Diameter P2P Networks. In *Proc. 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2001.

[182] C. Papadimitriou. Private Communication. In *Visit at ETH Zurich*, 2006.

[183] C. H. Papadimitriou. Algorithms, Games, and the Internet. In *Proc. 33rd ACM Symposium on Theory of Computing (STOC)*, pages 749–753, 2001.

[184] A. Pasick. File-sharing Network Thrives Beneath the Radar. In *Yahoo! News*, 2004.

[185] R. Pastor-Satorras and A. Vespiagnani. Immunization of Complex Networks. *Physical Review Letter*, 65, 2002.

[186] B. Patt-Shamir. A Note on Efficient Aggregate Queries in Sensor Networks. *Theor. Comput. Sci.*, 370(1-3):254–264, 2007.

[187] D. Peleg. *Distributed Computing: A Locality-sensitive Approach.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[188] D. Peleg and E. Upfal. The Token Distribution Problem. *SIAM Journal on Computing*, 18(2):229–243, 1989.

[189] R. Peterson and E. G. Sirer. Going Beyond Tit-for-Tat: Designing Peer-to-Peer Protocols for the Common Good. In *Proc. Workshop on Future Directions in Distributed Computing*, 2007.

[190] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkatarami. Do Incentives Build Robustness in BitTorrent? In *Proc. 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2007.

[191] C. Plaxton, R. Rajaraman, and A. W. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed Environment. In *Proc. 9th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 311–320, 1997.

[192] C. G. Plaxton. Load Balancing, Selection and Sorting on the Hypercube. In *Proc. 1st Ann. ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 64–73, 1989.

[193] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed Environment. In *Proc. 9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 311–320, 1997.

[194] PPLive. Website. http://www.pplive.com/.

[195] Y. Qiao and F. E. Bustamante. Structured and Unstructured Overlays Under the Microscope – A Measurement-based View of Two P2P Systems that People Use. In *Proc. USENIX Annual Technical Conference*, 2006.

[196] D. Qiu and R. Srikant. Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Systems. In *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2004.

[197] J. S. Rakoff. Copyright Law and the Internet. *NYSBA Entertainment, Arts and Sports Law Journal*, 14(2), 2003.

[198] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content-Addressable Network. In *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 161–172, 2001.

[199] S. Rhea, B.-G. Chun, J. Kubiatowicz, and S. Shenker. Fixing the Embarrassing Slowness of OpenDHT on PlanetLab. In *Proc. 2nd Conference on Real, Large Distributed Systems (WORLDS)*, 2005.

[200] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling Churn in a DHT. In *Proc. USENIX Ann. Technical Conference*, 2004.

[201] RIAA. http://www.riaa.com/issues/piracy/default.asp. Issues: Anti-Piracy.

[202] A. Rosén. A Note on Models for Non-Probabilistic Analysis of Packet Switching Networks. *Inf. Process. Lett.*, 84(5):237–240, 2002.

[203] A. Roth. The Price of Malice in Linear Congestion Games. In *Under submission*, 2008.

[204] T. Roughgarden. Stackelberg Scheduling Strategies. In *Proc. 33rd ACM Symposium on Theory of Computing (STOC)*, pages 104–113, 2001.

[205] T. Roughgarden. *Selfish Routing and the Price of Anarchy.* MIT Press, 2005.

[206] T. Roughgarden and E. Tardos. How Bad is Selfish Routing? *Journal ACM*, 49(2), 2002.

[207] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. IFIP/ACM Int. Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[208] J. Saia, A. Fiat, S. Gribble, A. Karlin, and S. Saroiu. Dynamically Fault-Tolerant Content Addressable Networks. In *Proc. 1st Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[209] S. Sanghavi and B. Hajek. A New Mechanism for the Free-rider Problem. In *Proc. 3rd ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, 2005.

[210] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proc. Multimedia Computing and Networking (MMCN)*, 2002.

[211] C. Scheideler. Models and Techniques for Communication in Dynamic Networks. In *Proc. 19th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 27–49, 2002.

[212] C. Scheideler. How to Spread Adversarial Nodes? Rotate! In *Proc. 37th ACM Symposium on Theory of Computing (STOC)*, 2005.

[213] C. Schindelhauer and G. Schomaker. Weighted Distributed Hash Tables. In *Proc. 17th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 2005.

[214] S. Sen and J. Wang. Analyzing Peer-to-Per Traffic Across Large Networks. *IEEE/ACM Transactions on Networking*, 12(2), 2003.

[215] B. A. Shirazi, K. M. Kavi, and A. R. Hurson. *Scheduling and Load Balancing in Parallel and Distributed Systems*. IEEE Computer Science Press, 1995.

[216] J. Shneidman and D. C. Parkes. Rationality and Self-Interest in Peer to Peer Networks. In *Proc. 2nd Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[217] J. Shneidman, D. C. Parkes, and L. Massoulié. Faithfulness in Internet Algorithms. In *Proc. Workshop on Practice and Theory of Incentives and Game Theory in Networked Systems (PINS)*, 2004.

[218] R. Shostak, M. Pease, and L. Lamport. Reaching Agreement in the Presence of Faults. *Journal ACM*, 27(2):228–234, 1980.

[219] A. Singh, T.-W. J. Ngan, P. Druschel, and D. S. Wallach. Eclipse Attacks on Overlay Networks: Threats and Defenses. In *Proc. 25th Annual IEEE Conference on Computer Communications (INFOCOM)*, 2006.

[220] Skype. Website. http://www.skype.com/.

[221] Y. Sovran, A. Libonati, and J. Li. Pass It On: Social Networks Stymie Censors. In *Proc. 7th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2008.

[222] T. K. Srikant and S. Toueg. Simulating Authenticated Broadcasts to Derive Simple Fault-Tolerant Algorithms. *Journal of Distributed Computing*, 2(2):80–94, 1987.

[223] M. Steiner, E. W. Biersack, and T. Ennajjary. Actively Monitoring Peers in KAD. In *Proc. 6th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2007.

[224] R. Steinmetz and Klaus Wehrle (Eds). *Peer-to-Peer Systems and Applications.* Springer, 2005.

[225] R. Stevens and G. R. Wright. *TCP/IP Illustrated Vol. 2 (The Implementation).* Addison-Wesley, 1995.

[226] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2001.

[227] D. Stutzbach and R. Rejaie. Understanding Churn in Peer-to-Peer Networks. In *Proc. 6th Internet Measurement Conference (IMC)*, 2006.

[228] D. Stutzback and R. Rejaie. Improving Lookup Performance over a Widely-Deployed DHT. In *Proc. 25th Annual IEEE Conference on Computer Communications (INFOCOM)*, 2006.

[229] R. Subramanian and Brian D. Goodman (Eds). *Peer-to-Peer Computing: The Evolution of a Disruptive Technology.* IGI Global, 2005.

[230] X. Sun, R. Torres, and S. Rao. Preventing DDoS Attacks with P2P Systems through Robust Membership Management. *Technical Report TR-ECE-07-13, Purdue University*, 2007.

[231] K. Tamilmani, V. Pai, and A. Mohr. SWIFT: A System with Incentives for Trading. In *Proc. 2nd Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, 2004.

[232] J. Tian and Y. Dai. Understanding the Dynamic of Peer-to-Peer Systems. In *Proc. 6th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2007.

[233] C. A. Tovey. A Simplified NP-Complete Satisfiability Problem. *Discrete Applied Mathematics*, 8:85–89, 1984.

[234] J. Travers and S. Milgram. An Experimental Study of the Small World Problem. *Sociometry*, 32(4):425–443, 1969.

[235] R. Van Renesse, K. P. Birman, and W. Vogels. Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining. *ACM Transactions on Computing Systems*, 21(2):164–206, 2003.

[236] R. van Renesse and A. Bozdog. Willow: DHT, Aggregation, and Publish/Subscribe in One Protocol. In *Proc. 3rd Int. Workshop on Peer-To-Peer Systems (IPTPS)*, 2004.

[237] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. KARMA: A Secure Economic Framework for P2P Resource Sharing. In *Proc. 1st Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, 2003.

[238] J. von Neuman. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928.

[239] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, 1944.

[240] D. S. Wallach. A Survey of Peer-to-Peer Security Issues. In *Int. Symposium on Software Security*, 2002.

[241] W. Wang and B. Li. Market-driven Bandwidth Allocation in Selfish Overlay Networks. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*, pages 36–46, 2005.

[242] J. L. Welch and N. Lynch. A New Fault-Tolerant for Clock-Synchronization. *Information and Communication*, 77:1–36, 1988.

[243] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A Lightweight Network Location Service Without Virtual Coordinates. In *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2005.

[244] Xing. Website. http://www.xing.com/.

[245] M. Yang, H. Chen, B. Y. Zhao, Y. Dai, and Z. Zhang. Deployment of a Large Scale Peer-to-Peer Social Network. In *Proc. 1st Workshop on Real, Large Distributed Systems*, 2004.

[246] A. C. Yao. Protocols for Secure Computations. In *Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1982.

[247] Zattoo. Website. http://www.zattoo.com/.

[248] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A Resilient Global-Scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, 2003.

Curriculum Vitae

July 31, 1978	Born in Hedingen, Switzerland
1985–1991	Primary and secondary school in Hedingen
1991–1998	High school (Matura type B) at the Kantonsschule Limmattal, Urdorf, Switzerland
1999–2004	Studies in computer science, ETH Zurich, Switzerland *Minor subject:* Micro and macro economics *Internship:* CERN, Geneva
September 2004	MSc in computer science, ETH Zurich, Switzerland
2004 – 2008	PhD student, research and teaching assistant, Distributed Computing Group, Prof. Dr. Roger Wattenhofer, ETH Zurich, Switzerland
April 2008	PhD defense and PhD degree, Distributed Computing Group, ETH Zurich, Switzerland *Advisor:* Prof. Dr. Roger Wattenhofer *Co-examiners:* Prof. Dr. Boaz Patt-Shamir Tel Aviv University, Tel Aviv, Israel Prof. Dr. Tim Roughgarden Stanford University, California, USA

Publications

The following lists all publications written during the roughly three and a half years of my PhD time at ETH Zurich. Note that—due to space constraints and in order to have a consistent subject—only a subset of these papers are presented in this thesis, namely the papers focusing on the peer-to-peer dynamics and cooperation challenge. In Part I, Chapter 3 is based on the papers at IPTPS 2005 and IWQoS 2006, and Chapter 4 is based on the paper at HiPC 2006 (an extension for wireless networks is studied in the paper at WICON 2006). In Part II, the BitThief chapter (Chapter 8) has been published at HotNets 2006, Chapter 9 has been published at IPTPS 2006 and PODC 2006, Chapter 10 also at PODC 2006, and Chapter 11 at EC 2008. Finally, some results of this thesis have not been published yet.

1. *Distributed Computation of the Mode.* Fabian Kuhn, Thomas Locher, and Stefan Schmid. 27th ACM Symposium on the Principles of Distributed Computing (PODC), Toronto, Canada, August 2008.

2. *Tight Bounds for Delay-Sensitive Aggregation.* Yvonne Anne Oswald, Stefan Schmid, and Roger Wattenhofer. 27th ACM Symposium on the Principles of Distributed Computing (PODC), Toronto, Canada, August 2008.

3. *On the Windfall of Friendship: Inoculation Strategies on Social Networks.* Dominic Meier, Yvonne Anne Oswald, Stefan Schmid, and Roger Wattenhofer. ACM Conference on Electronic Commerce (EC), Chicago, Illinois, USA, July 2008.

4. *Distributed Disaster Disclosure.* Bernard Mans, Stefan Schmid, and Roger Wattenhofer. 11th Scandinavian Workshop on Algorithm Theory (SWAT), Gothenburg, Sweden, July 2008. Springer Lecture Notes in Computer Science, LNCS 5124.

5. *Distributed Asymmetric Verification in Computational Grids.* Michael Kuhn, Stefan Schmid, and Roger Wattenhofer. 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS), Miami, Florida, USA, April 2008.

6. *Structuring Unstructured Peer-to-Peer Networks.* Stefan Schmid and Roger Wattenhofer. 14th Annual IEEE International Conference on

High Performance Computing (HiPC), Goa, India, December 2007. Springer Lecture Notes in Computer Science, LNCS 4873.

7. *Manipulation in Games.* Raphael Eidenbenz, Yvonne Anne Oswald, Stefan Schmid, and Roger Wattenhofer. 18th International Symposium on Algorithms and Computation (ISAAC), Sendai, Japan, December 2007. Springer Lecture Notes in Computer Science, LNCS 4835.

8. *Push-to-Pull Peer-to-Peer Live Streaming.* Thomas Locher, Remo Meier, Stefan Schmid, and Roger Wattenhofer. 21st International Symposium on Distributed Computing (DISC), Lemesos, Cyprus, September 2007. Springer Lecture Notes in Computer Science, LNCS 4731.

9. *Rescuing Tit-for-Tat with Source Coding.* Thomas Locher, Stefan Schmid, and Roger Wattenhofer. 7th IEEE International Conference on Peer-to-Peer Computing (P2P), Galway, Ireland, September 2007.

10. *Mechanism Design by Creditability.* Raphael Eidenbenz, Yvonne Anne Oswald, Stefan Schmid, and Roger Wattenhofer. 1st International Conference on Cominatorial Optimization and Applications (COCOA), Xi'an, Shaanxi, China, August 2007. Springer Lecture Notes in Computer Science, LNCS 4616.

11. *Dynamic Internet Congestion with Bursts.* Stefan Schmid and Roger Wattenhofer. 13th Annual IEEE International Conference on High Performance Computing (HiPC), Bangalore, India, December 2006. Springer Lecture Notes in Computer Science, LNCS 4297.

12. *Free Riding in BitTorrent is Cheap.* Thomas Locher, Patrick Moor, Stefan Schmid, and Roger Wattenhofer. 5th Workshop on Hot Topics in Networks (HotNets), Irvine, California, USA, November 2006.

13. *Cryptree: A Folder Tree Structure for Cryptographic File Systems.* Dominik Grolimund, Luzius Meisser, Stefan Schmid, and Roger Wattenhofer. 25th IEEE Symposium on Reliable Distributed Systems (SRDS), Leeds, United Kingdom, October 2006.

14. *eQuus: A Provably Robust and Locality-Aware Peer-to-Peer System.* Thomas Locher, Stefan Schmid, and Roger Wattenhofer. 6th IEEE International Conference on Peer-to-Peer Computing (P2P), Cambridge, United Kingdom, September 2006.

15. *A TCP with Guaranteed Performance in Networks with Dynamic Congestion and Random Wireless Losses.* Stefan Schmid and Roger Wattenhofer. 2nd International Wireless Internet Conference (WICON), Boston, Massachusetts, USA, August 2006.

16. *When Selfish Meets Evil: Byzantine Players in a Virus Inoculation Game.* Thomas Moscibroda, Stefan Schmid, and Roger Wattenhofer. 25th ACM Symposium on the Principles of Distributed Computing (PODC), Denver, Colorado, USA, July 2006.

17. *On the Topologies Formed by Selfish Peers.* Thomas Moscibroda, Stefan Schmid, and Roger Wattenhofer. 25th ACM Symposium on the Principles of Distributed Computing (PODC), Denver, Colorado, USA, July 2006.

18. *A Blueprint for Constructing Peer-to-Peer Systems Robust to Dynamic Worst-Case Joins and Leaves.* Fabian Kuhn, Stefan Schmid, Joest Smit, and Roger Wattenhofer. 14th IEEE International Workshop on Quality of Service (IWQoS), Yale University, New Haven, Connectitut, USA, June 2006.

19. *Havelaar: A Robust and Efficient Reputation System for Active Peer-to-Peer Systems.* Dominik Grolimund, Luzius Meisser, Stefan Schmid, and Roger Wattenhofer. 1st Workshop on the Economics of Networked Systems (NetEcon), University of Michigan, Ann Arbor, Michigan, USA, June 2006.

20. *Algorithmic Models for Sensor Networks.* Stefan Schmid and Roger Wattenhofer. 14th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS), Island of Rhodes, Greece, April 2006. An extended version of this paper will appear as a book chapter in *Algorithms and Protocols for Wireless Sensor Networks* (Editor: Azzedine Boukerche), John Wiley & Sons Inc, ISBN: 0471798134, expected publication date: October 2008.

21. *On the Topologies Formed by Selfish Peers.* Thomas Moscibroda, Stefan Schmid, and Roger Wattenhofer. 5th International Workshop on Peer-to-Peer Systems (IPTPS), Santa Barbara, California, USA, February 2006.

22. *A Robust Interference Model for Wireless Ad-Hoc Networks.* Pascal von Rickenbach, Stefan Schmid, Roger Wattenhofer, and Aaron Zollinger. 5th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN), Denver, Colorado, USA, April 2005.

23. *A Self-Repairing Peer-to-Peer System Resilient to Dynamic Adversarial Churn.* Fabian Kuhn, Stefan Schmid, and Roger Wattenhofer. 4th International Workshop on Peer-to-Peer Systems (IPTPS), Cornell University, Ithaca, New York, USA, February 2005. Springer Lecture Notes in Computer Science, LNCS 3640.

24. *Parallel Compilation of CMS Software.* Shaun Ashby, Giulio Eulisse, Stefan Schmid, and Lassi Tuura. Computing in High Energy and Nuclear Physics Conference (CHEP), Interlaken, Switzerland, September 2004.

Series in Distributed Computing
Edited by Roger Wattenhofer

Peer-to-peer (p2p) computing is one of the most intriguing new networking paradigms. At the heart of the paradigm lies the idea of leveraging the resources of the system's participants. Thus, potentially scalable and robust architectures can be built. However, making use of the decentralized resources is challenging. The peers are under the control of the individual users who may only connect to the network for a short period of time. Consequently, there are frequent membership changes and p2p systems are highly dynamic. Peer-to-peer solutions are also faced with the fact that it is not always in the (anonymous) users' interest to contribute their resources. Rather, a user may seek to exploit the system without reciprocating.

This volume studies the challenges of the dynamics and cooperation in p2p computing. Algorithms for p2p systems are presented that maintain desirable network properties in dynamic environments. Our measurement study of BitTorrent – one of the most traffic intensive applications on the Internet – shows that today's peer-to-peer networks still fail to fend off uncooperative peers. The game-theoretic analysis of unstructured p2p networks indicates that both the performance and the stability of a system can suffer severely in case of selfish behavior. Finally, we introduce a new mathematical framework which allows us to evaluate a system's robustness to malicious attacks and which is also useful for the analysis of social networks. The theoretic findings are complemented by a case study which identifies vulnerabilities in the popular Kad network.

About the author:

Stefan Schmid received his MSc degree in computer science from the Swiss Federal Institute of Technology (ETH), Zurich, Switzerland in 2004. In the same year, he joined the Distributed Computing Group of Professor Roger Wattenhofer at ETH Zurich as a PhD student and teaching assistant. In 2008, he earned his PhD degree for his work on algorithmic challenges in peer-to-peer computing.

Hartung-Gorre Verlag Konstanz

ISSN 1861-1591   ISBN-10: 3-86628-205-2   ISBN-13: 978-3-86628-205-6   € 64.–

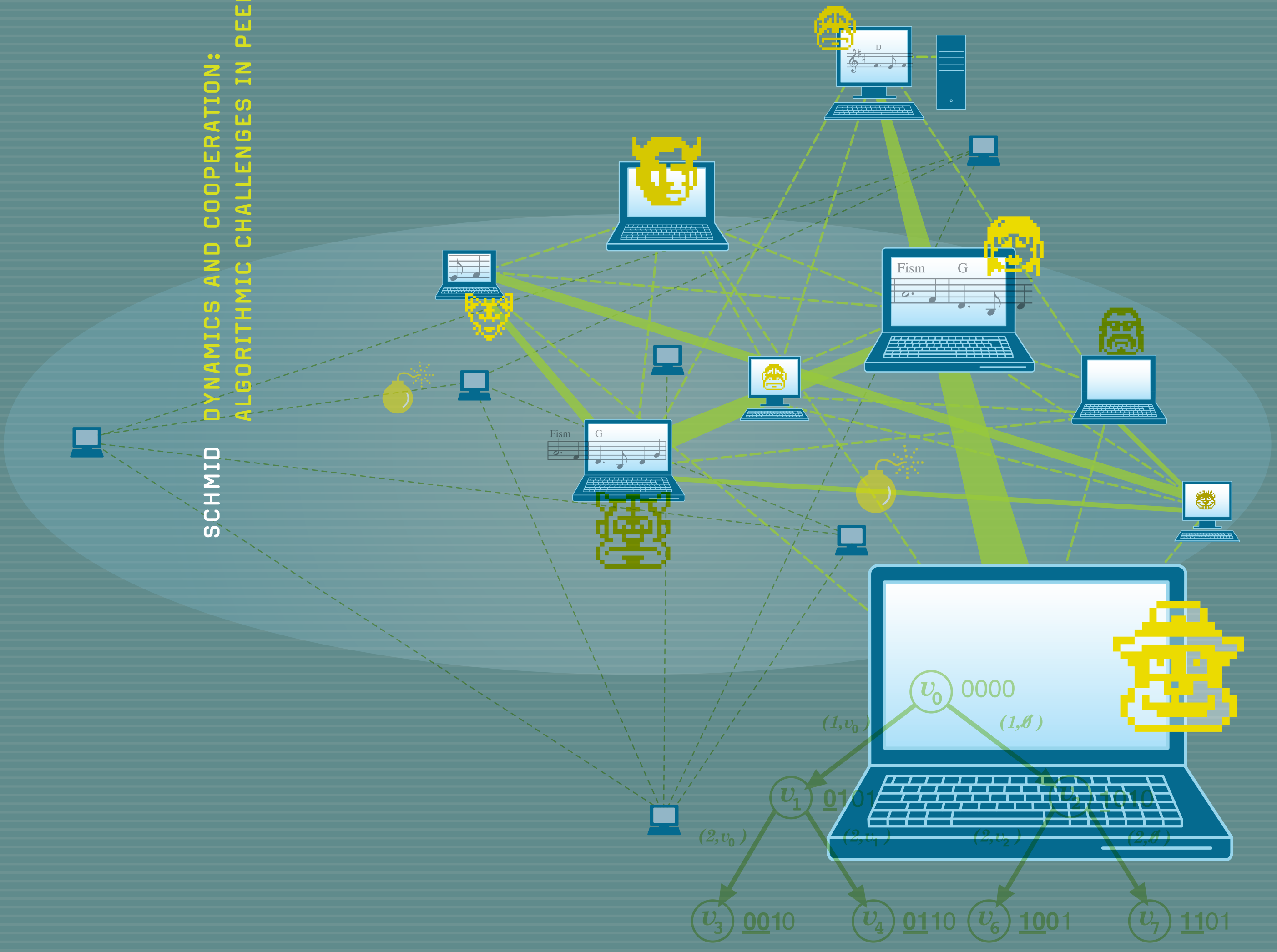
SCHMID   DYNAMICS AND COOPERATION:
ALGORITHMIC CHALLENGES IN PEER-TO-PEER COMPUTING

Series in Distributed Computing   Volume 6   2008

DYNAMICS AND COOPERATION

ALGORITHMIC CHALLENGES IN PEER-TO-PEER COMPUTING

Stefan Schmid



Hartung-Gorre Verlag Konstanz