Peer-to-Peer System Design:
A Socioeconomic Approach

Rameez Rahman

Peer-to-Peer System Design:
A Socioeconomic Approach

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 6 september 2011 om 12:30 uur

door **Rameez RAHMAN**

Master of Science, University of Essex
geboren te Karachi, Pakistan

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. H.J. Sips

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof.dr.ir. H.J. Sips	Technische Universiteit Delft, promotor
Dr.ir. Johan Pouwelse	Technische Universiteit Delft, copromotor
Prof.dr. F. Brazier	Technische Universiteit Delft
Prof.dr.ir. M.R. van Steen	Vrije Universiteit Amsterdam
Prof.dr. ir. J.A. La Poutre	Universiteit Utrecht
Prof. Alberto Montresor	Università degli Studi di Trento, Italia
Prof. Pedro Garcia Lopez	Universitat Rovira i Virgili, España
Prof.dr.ir. G.J.P.M. Houben	Technische Universiteit Delft, reservelid

Published and distributed by: Rameez Rahman
E-mail: rrameez@gmail.com

ISBN: 978-90-79982-10-3

Keywords: Peer-to-Peer systems, BitTorrent, Incentives, Design Space Analysis, Modeling, Simulation, Game Theory, Credit Crunch.

Printed in The Netherlands by: Wöhrmann Print Service.

This work was supported by the Higher Education Commission of Pakistan (HEC), and also by the Future and Emerging Technologies programme FP7-COSI-ICT of the European Commission through the QLectives project (grant no.: 231200) and the P2P-Next project (grant no.:216217).

Acknowledgements

I would like to begin by thanking my mentor David Hales. **Dave**, the virtues of this work are due to you, and the vices due to me. Your influence, even when I was rebeling against it, can be discerned throughout this thesis. I cannot pay a greater tribute to you than by admitting that from my two favorite chapters in this thesis, one was written (in my mind) against you (Chapter 6), while the other was written for you (Chapter 7). During the good times and the bad, whether we were engaged in mutual backslapping or mutual insults, I was always struck by your generosity, your breadth of intellect and your sincere struggle against apathy. You truly are a special person. Thank you so much!

Henk, I want to thank you for your kindness and your magnanimity. You have always supported me and I would not have had such an easy time were it not for your comforting shadow. I am also very grateful that despite being such an accomplished academic, you never tried to impose your views upon me. You never pressurized me to change my working style to conform to your standards, and you always encouraged me when I came up with new ideas. Thank you also for taking care of the so many administrative issues that kept creeping up over the years.

Johan, full credit is due to you for assembling such a nice group of people on the 9th floor. I hope you maintain your position and keep bringing in grants, and wonderful people, to our floor. It has been an eventful time with you and I want to thank you for giving me the opportunity of working here.

Michel, thank you so much for everything. Your guidance and support at all stages have been very helpful. You think and feel so deeply about things, and that inspires others around you to be positive and wholesome. You are a treasure to your friends: gifted though not very ambitious, wise but not clever, meticulous yet not conservative. Spiritually, I always felt that you were *Pierre* to my *Andrei*, and we complemented each other very well. I also want to thank Apple for being a very dear friend to Sidra. And of course best wishes for Ananda.

Tamas, thanks a lot for supporting me through all my low phases. It has been quite a journey. I believe that despite all the challenges, we managed to make a very good time of it. You are one of the most honest and decent people I have met and you have inspired me to live up to your high standards. Thank you especially for all the nice, long talks: I would not have survived without them! Thank you also for your humility and modesty, which allowed me to discuss technical matters with you without hesitation. And yes, do

keep the faith. All the best wishes for Marci and Barnus.

Ali, you are a special friend. I am grateful for the support that I have always received from you. Also, on current news and affairs, you are the most well-informed friend that I have, and it is always nice to learn new stuff from you. I wish you all the best for your thesis.

Arno, I shall never forget your kind gestures, both in times of grief and joy. Thank you! You are a true gentleman.

Naza, thank you for being a breath of fresh air. I am grateful to you for sharing your acute insights with us and for giving us a touch of Brazilian liveliness during dull times.

Lucia, thank you for your encouragement and for being a comrade-in-arms. I wish you and **Jelle** all the best for the future. Also, I wish you luck for your thesis (even though you do not need it).

Alexandru and **Ana**, thank you for making the initial days nice and enjoyable. It was wonderful knowing you both.

Boxun, thank you for keeping my connection to China alive and for sharing, among other things, videos of Chinese cartoons that we grew up watching. Thanks also for all the nice badges. Hope to meet you in Beijing some day in the future, perhaps on a boat near the *TianTan*.

Mihai, thank you for always being very kind to me and forgiving my follies. All the best in designing the new Internet!

Nitin, conversing with you was always a joy. World cup 2011 would not have been so enjoyable had it not been for the frenzied *gtalk* messages that we kept exchanging, sometimes over-by-over. I wish you all the best for your future.

Adele, thank you for being such a nice colleague and research partner. Working with you has been a joy. Hope to see you as a professor in Beijing or Tsinghua University some day soon!

Victor, thank you for being the 'unique eccentric scientist' figure. Looking forward to witnessing great things from you in the future.

Rahim, thank you for the nice talks, especially in the gym, and good luck for your PhD and your life.

Riccardo, thank you for being super cool!

Niels, you are the first student that became a colleague, a nice trend that I hope will continue in the future.

Dimitra, I hope everything goes well with you and that you get published in a physics journal soon.

Boudewijn, thank you for being patient with us 'bad-programmers' and for always being very sharp and thorough.

Paulo, I want to thank you for all the DAS related support. All the best for your report. I also thank **Munire** and **Stephen** for the oft-needed technical support.

Thank you **Ilse, Rina, Esther** and **Monique** for all the hard work that you put in for making sure that the administrative tasks related to our research go smoothly. Your presence is a blessing.

Julio, you have been a wonderful friend. I cannot thank you enough for your support during the four years and I hope we remain this close in the future as well.

Boudewijn and **Yu**, thank you for your support over the years. Boudewijn, I can never forget your love at a time when I was nearly down and out. You are a great person, a great guide and a great friend. Hope to see you in Pakistan some day soon. All the best to both of you for your creative efforts in China.

I want to thank my sister, **Tabinda**, and my beautiful niece and nephew, **Sophie** and **Nirvan**, for waiting patiently for my return. Thank you *baji* for actively supporting my decision to pursue a PhD. This thesis would not have been possible without you. Thank you and I shall see you soon!

I would like to thank my grandmother **Dilafroz Qureshi**, for being the sweetest grandmother in the world.

I want to thank my parents **Ahfaz-ur-Rahman** and **Mahnaz Rahman**, for their inimitable love. If I am able to care for Myna half as much as you have cared for me, I would have done a great job. Thank you for all the big things and for all the little things. Thank you for taking care of us through thick and thin. Thank you for the wonderful humanistic teachings, the import of which is also evident in this technical thesis.

Contents

Chapter 1

Introduction

In this thesis, we aim to study the usage of socioeconomic ideas in peer-to-peer (P2P) systems. Following the advice of the late Stephen Jay Gould, we begin by clearly stating our underlying inclination and approach towards P2P systems. We approach P2P systems as digital workplaces, as economies, and as societies, which can serve as test-beds for experimenting with novel approaches to socioeconomic problems. Having said that, this thesis is practically grounded: we propose solutions that deal with problems in existing P2P systems, and do not conjure up solutions to irrelevant problems.

Hence this thesis is 'one long argument' for the evaluation of various socioeconomic approaches–some run of the mill and ordinary, some radical and revolutionary–for solving a host of problems in P2P systems. In doing so, we have tried to challenge conventional wisdom where possible.

On one level we present solutions that are practical and can easily be applied to existing systems, without requiring massive structural changes. On another level, we examine the usage of recurring concepts such as "rationality", "free-riding" and "social welfare", etc., in the literature, and argue for the consideration of alternate viewpoints. On yet another level we argue for re-examining the dominant use of game theoretical solution concepts for protocol evaluation, and also propose complementary approaches. Finally, we also identify and open up new problem areas that could properly be described as the macroeconomic problems of P2P communities, and present approaches that can alleviate those problems.

In some sense, then, this thesis is a microcosm of various larger debates not limited to the P2P arena, including between proponents of rational choice theory and those who argue for alternate frameworks; between advocates of analytical modeling and those who argue for increased usage of a simulation based methodology for modeling and understanding complex systems; between advocates of simple, practical designs as opposed to heavyweight 'fool-proof' systems; and finally, this perhaps the most ill-defined of all but real nevertheless, between those who approach P2P systems as traditional systems and those who regard them as social systems used by actual people with different resources,

interests, backgrounds and needs.

We note that in so far as our work touches upon the afore-mentioned debates, we claim to be neither neutral nor partial. In fact, in the best engineering tradition, we act as opportunists, making use of whatever makes sense for the problem at hand!

1.1 A brief history of P2P Systems

The P2P 'phenomenon' started with the *Napster* file-sharing protocol [85], which was followed by many others such systems. Napster was a breakthrough phenomenon and gained notoriety upon being brought to court in a lawsuit filed by several major record labels. In its heyday, Napster had several millions of users. After Napster was shut down, several clients emerged that also catered to a large number of users including *Kazaa*, *eDonkey*, etc. The popular *Gnutella* network appeared in early 2000. Subsequently, many clients such as *Limewire* and *Morpheus* based on the Gnutella protocol emerged. These clients at various times boasted tens of millions of users.

At the same time, other P2P applications, not limited to file-sharing also emerged. Technologies such as *Skype* and other *voice over ip* (VoIP) clients emerged as potential competitors to major telecommunications providers. Other P2P systems including caching systems such as *Dalesa* [49], search engines like *Yacy* [53], audio systems like *Spotify* [50] and even digital currencies like *Bitcoin* [47] have also emerged in recent years.

1.2 Grounding our work

In order to analyze the general trend on how socioeconomic concepts have been applied in P2P Systems, we ground our work in the realm of P2P file sharing systems. In particular, we focus on the following three systems: *The BitTorrent protocol*, *Private P2P file-sharing communities*, and *Tribler*. Next we give a brief overview of each.

1.2.1 BitTorrent

Around 2003, the popular BitTorrent protocol emerged [18]. Currently, it is the most widely used P2P protocol and generates huge volumes of Internet traffic. Earlier P2P file-sharing systems suffered from two major problems, namely: a) Lack of contribution by a majority of the users (*freeriding*) [1]; and b) Spread of malicious content by some peers (*Spam*) [69]. The BitTorrent protocol solved these problems that plagued other file-sharing systems, albeit in two different ways.

The BitTorrent protocol solved the 'freeriding' problem by building 'robust incentives' based on the Tit for Tat (TFT) strategy. TFT is the strategy that became popular from the tournament organized by Axelrod[1] in order to determine the best strategy for playing the Iterated Prisoner's Dilemma [7]. TFT in essence resembles the 'Reciprocal Altruism' model put forth by Trivers [125] to explain cooperation in biological organisms. In BitTorrent, using TFT peers upload to those others who reciprocate to them the most. This ensures that peers who upload less, get less in return.

BitTorrent solves the 'spam' problem by avoiding it: content location and dissemination are not part of the BitTorrent protocol. Metafiles called *torrents* contain information needed by a BitTorrent client to start its download. These torrents can be disseminated through any out of band means, such as through emails, on websites, etc. There are many centralized websites which publish torrents.

Torrents contain information about a central server called a *tracker* and an infohash of all the pieces of a file. A tracker acts as a bootstrap server and provides newly arriving peers with addresses of other peers. Peers who have a complete copy of the file are called *seeders*, while peers who do not yet have a complete copy are called *leechers*. The amalgam of seeders and leechers associated with a file is called a *swarm*. Leechers in a swarm get free pieces from the seeders, while they trade pieces with each other based on a rate based TFT, using which they prefer fastest uploading partners.

BitTorrent does not provide incentives for one crucial aspect, i.e., seeding. After completely downloading a file, a peer does not have any incentive to share its complete copy with the community. This problem of providing peers with seeding incentives has been addressed by many private BitTorrent communities, which we discuss next.

1.2.2 Private P2P Communities

As mentioned, since BitTorrent does not provide data location and dissemination services, several websites have emerged that publish torrents. In order to incentivize seeding, many websites maintain a private tracker which maintains centralized accounts of users and records their upload and download volumes. Such websites are called private BitTorrent communities. Such communities employ sharing ratio enforcement (SRE) and require all peers to maintain sharing ratios (upload / download) above a certain threshold. Peers who do not do so are either banned temporarily from downloading or expelled from the community. There are many private BitTorrent communities around e.g., TvTorrents [52], BitSoup [48], SuperNova [51], etc. Some of these use a credit based scheme, while others simply record the upload and download amounts and use these to calculate peers' sharing ratios.

[1] It was in their book, 'Prisoner's Dilemma' [104] that A. Rapoport and A. Chammah, first introduced the Tit-for-Tat Strategy. This fact usually goes unmentioned in the literature.

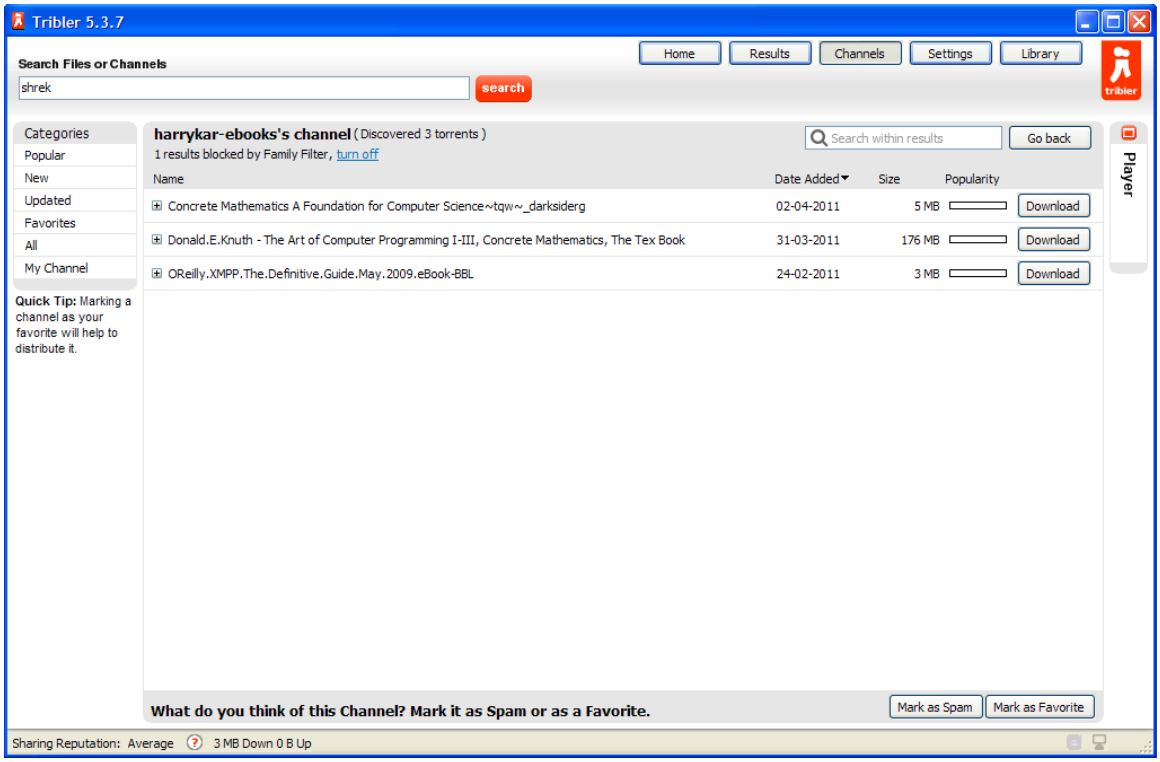


Figure 1.1: VoteCast deployed in Tribler.

1.2.3 Tribler

Tribler is a BitTorrent client, which provides the same service as private BitTorrent communities, albeit in a decentralized manner [96]. Tribler provides the dissemination of torrents and allows peers to search for them using decentralized mechanisms. Tribler also provides a reputation mechanism *BarterCast* to incentivize users to seed. Previously, Tribler did not have a mechanism using through peers could judge the quality of a piece of content. In Chapter 3, we describe *Votecast*, a decentralized method for determining quality (and fighting spam) in P2P communities, which we have implemented in Tribler. The lower part of Figure 1.1 shows voting options in the latest Tribler release.

1.3 The Four Pillars of P2P Systems

P2P systems, generally involve the participation of autonomous nodes called peers, who get together to pursue a common goal. We put forth that P2P Systems depend on roughly the following three pillars or dimensions, which are, in theory, all distributed: a) mechanisms for spreading information between peers; b) mechanisms for determining the quality/authenticity of the spread information; c) mechanisms for ensuring that nodes follow the prescribed protocol(s).

In practice, not all components that belong to these pillars are distributed or completely distributed, and usually require centralized solutions. For example, the bootstrap mechanism, which falls under pillar (a), used to inform newly arriving peers about other peers, is usually centralized. Most deployed P2P systems use a bootstrap server, which provides new arriving peers with information about existing, live peers in the network. Similarly, components under pillars (b) and (c) can be both centralized and distributed in practice. For example, the famous file-sharing P2P protocol BitTorrent implements an incentive mechanism (that falls under pillar (c)) to ensure that nodes follow the protocol, in a distributed manner. But on the other hand, private BitTorrent communities normally use a centralized component for incentivizing seeding, a crucial BitTorrent activity. Moderation of content (that falls under pillar (b)) is also executed centrally in private BitTorrent communities through the help of dedicated moderators, who mark content as 'high quality'.

We add a fourth, auxiliary pillar to the three pillars described above, which merits a place in this list owing to its pervasiveness in the literature: $d)$ mechanisms for the evaluation of robustness, performance and, fairness of P2P protocols.

1.4 Research Questions

In this thesis, we focus on pillars (b), (c), and (d), while pillar (a) is orthogonal to our concerns. We seek to examine the usage of socioeconomic ideas in P2P file sharing systems and determine: 1) whether unexplored areas (under pillars *b*,*c* and *d*) could benefit from their usage; 2) whether existing approaches could benefit from extensions and modifications; and the last, more radical, 3) whether existing approaches could be replaced by an alternate set of approaches and ideas, which can lead to improvements. For accomplishing this, we seek to answer the following research questions:

Can a decentralized, socially inspired mechanism be used to determine content quality in P2P Systems? The evaluation of information about peers and content, is an integral aspect of P2P systems. It can determine the extent to which peers are incentivized to contribute to the system; able to recognize the contribution of their peers; and finally, able to search for 'right', high quality content. Devising mechanisms to ensure that peers are provided with accurate information about content quality can end up being heavyweight in terms of bandwidth overhead and design complexity. It is therefore, a worthwhile idea to invest efforts into the design of lightweight mechanisms that can be deployed in practical systems.

What are the macroeconomic problems of BitTorrent Communities? Many private BitTorrent communities apply credit based enforcement or sharing ratio enforcement (SRE) through which they enforce contribution from peers. It has been demonstrated that these mechanisms lead to increased supply as compared to communities that do not em-

ploy any such mechanisms [16, 71, 137]. On the other hand, anecdotal evidence suggests that while private communities do provide high download speeds due to oversupply, peers usually tend to have an extremely hard time earning credit or sharing ratio. Therefore, it is important to study why this happens and analyze the macroeconomic problems of such communities.

What effect does the role of user behavior and credit flow have in P2P Communities? Usually incentive mechanisms for private communities are devised taking into account one specific user model. However, users that take part in such communities have unequal resources and also exhibit different behavior. Thus, it is worthwhile to model different user behaviors and study their effects on the underlying incentive scheme. It is also relevant, in credit based incentive schemes, to analyze the effects of taxation on the well-being of peers.

Can P2P Systems benefit from the application of alternate economic visions? Most incentive schemes in P2P Systems have been devised under the frameworks supplied by mainstream economics. Since other viewpoints or economic visions have not been explored, we believe it is high time to do that. Participatory Economics (Parecon) [2] is one such economic system that has been proposed and fleshed out, but has not seen much experimentation. Parecon claims that it is more efficient and fair as compared to our existing systems. We would like to analyze the effects of applying Parecon inspired strategies in P2P systems, on system efficiency and fairness.

Can the robustness and performance of distributed protocols be evaluated using tools other than game theory? After a new protocol has been designed, a designer usually uses solution concepts from game theory in order to assess the robustness of the protocol. Due to the predictive powers and general applicability of game theory, system designers have used it in a variety of contexts. However they have not paid much attention to the development of alternate methods that could overcome limitations of a game-theoretic approach. For example, many many variants that exploit a protocol are devised after the protocol has been proved to be robust using a game-theoretic approach. Given this, we believe it is of primary importance to devise complementary techniques that could help designers in assessing the properties of their protocols more comprehensively.

1.5 Contributions and thesis outline

Our contributions are:

Overview of socioeconomic ideas in P2P Systems (Chapter 2) Before embarking on the identification of new problem areas and the proposal of novel solutions, it is worthwhile to do an overview of how socioeconomic ideas have been employed in the literature. In this chapter, we present an in-depth overview that will serve to guide and shape

the subsequent chapters.

Robust Vote Sampling in P2P Systems (Chapter 3) The explosion of freely available media content through BitTorrent file sharing networks over the Internet means that users need guides or recommendations to find the right, high quality content. Current systems rely on centralized servers to aggregate, rate, and moderate metadata for this purpose. We present the design and simulations, using real BitTorrent traces, for a method combining fully decentralized metadata dissemination, vote sampling, and ranking for deployment in the Tribler.org BitTorrent media client. Our design provides robustness to spam attacks, where metadata does not reflect the content it is attached to, by controlling metadata spreading and by vote sampling based on a collusion proof *experience* function. Our design is light-weight, fully decentralized, and offers good performance and robustness under realistic conditions. This chapter is largely based on the work published in [98].

BitCrunch: Credit Squeeze in private BitTorrent Communities (Chapter 4) Much BitTorrent activity takes place within private virtual communities called "Private Trackers" - a server that allows only community members to share files. Many private trackers implement "ratio enforcement" where the tracker monitors the upload and download behaviour of peers. If a peer downloads substantially more than it uploads then service is terminated. Tracker policies related to credit effect the performance of the community as a whole. We identify the possibility of a "credit squeeze" in which performance is reduced due to lack of credit for some peers. We consider statistics from a popular private tracker and results from a simple model (called "BitCrunch"). This chapter is largely based on the work published in [43].

Sustainable credit dynamics in P2P Communities (Chapter 5) Many peer-to-peer file sharing communities implement credit policies to provide incentives to users to contribute upload resources. Such policies implicitly assume a user model, i.e., how the user controlling each peer behaves. We show using an agent-based model that credit policies, based on bandwidth contribution, and a *selfish* user model, can lead to both "crunches" and "crashes" where the system seizes completely due to too little credit or too much credit. We explore the conditions that lead to these system pathologies and present a theoretical analysis that allows us to determine if a community is sustainable or will eventually crunch or crash. Finally we apply the analysis to produce a novel adaptive credit system that automatically adjusts credit policies to maintain sustainability. This chapter is largely based on the work published in [99].

Improving Efficiency and Fairness using Effort-based incentives (Chapter 6) Most P2P systems that have some kind of incentive mechanism reward peers according to their contribution, i.e. total bandwidth offered to the system. Due to the disparity in bandwidth capacity between P2P users on the Internet, the common effect of such mechanisms is that the fastest peers reap the highest benefits. We take a different approach

and study how to incentivize cooperation in P2P systems based on effort, i.e. contribution relative to capacity. We make the following contributions: 1) we argue that contribution-based incentive schemes in P2P systems unnecessarily disfavor slow peers and decrease overall system performance; 2) we advocate the use of principles from an alternative economic vision, Participatory Economics (Parecon), to inspire systems to be fair and to ensure maximization of the social welfare while being efficient at the same time, and 3) we present the results of simulations in which we apply principles from Parecon to two popular real life systems: a) the popular file sharing BitTorrent protocol; b) a generic credit based sharing ratio enforcement scheme. Our approach yields a higher system performance and fairness and offers new insights into P2P incentive design. This chapter is largely based on the work published in [100].

Design Space Analysis for Modeling Incentives in Distributed Systems (Chapter 7) Distributed systems without a central authority, such as peer-to-peer (P2P) systems, employ incentives to encourage nodes to follow the prescribed protocol. Game-theoretic analysis is often used to evaluate incentives in such systems. For a tractable analysis of complex systems, a game-theoretic approach requires a high level of abstraction of the design space. It follows that different designers can choose different abstractions to reach equally valid but contradictory results, a point that we demonstrate in this chapter in the context of the P2P protocol BitTorrent. To complement game-theoretic analysis, we propose a simulation-based method for modeling incentives, which we call *Design Space Analysis* (DSA). DSA provides a tractable analysis of competing protocol variants within a detailed design space. We apply DSA to P2P file swarming systems. With extensive simulations we analyze a wide-range of protocol variants and gain insights into their robustness and performance. To validate these results and to demonstrate the efficacy of DSA, we modify an instrumented BitTorrent client and evaluate protocols discovered using DSA. We show that they yield higher system performance and robustness relative to the reference implementation. This chapter is largely based on the work published in [101][2].

[2]The experiments for Chapter 7 were performed on the *Distributed ASCI Supercomputer 3* (http://www.cs.vu.nl/das3/)

Chapter 2

Overview of socioeconomic ideas in P2P systems

In this chapter we present an overview of how some socioeconomic ideas have been employed in the literature. P2P systems opened up the possibility of using enormous computing potential and content availability to a large number of users. However, because these systems are decentralized and not owned by anyone in particular, they also pose many challenges, some of which are in effect socioeconomic challenges masqueraded as technical challenges. There are two major challenges, one economic and the other social.

The economic problem can be described to be what is called the 'free-rider' problem. Free-riding is the behavior where peers make use of a resource without contributing anything in return. P2P systems function as non-excludable goods, i.e., public goods, and therefore can suffer from free-riding behavior.

The 'social' problem, if one likes, are the deliberate hostile actions of peers undertaken in order to either game the system to gain some benefits, or to actively harm it. This can include, for example, finding loopholes within the system or deliberately spreading false information, etc.

It can be discerned that both problems are intimately related and their solution can be loosely described as the provision of social and economic incentives to peers for following the protocol; abstaining from malicious activities; and contributing their resources.

2.1 Rationality: The Emergence of the Dominant Paradigm

A lot of work has been done on studying freeriding and incentivizing cooperation in P2P systems. Early peer to peer studies [37,109] documented that the popular P2P systems had rampant freeriders. Freeriding by people using the system suggested that they were not using the software as it was intended to be used by its designers, or to put it formally, users

were not being 'faithful' to the algorithms and protocols [115]. Two important works in P2P [26, 113] proposed that the view where peers were only regarded as obedient (always following the protocol), was insufficient. Borrowing from traditional economics and game theory, they proposed that nodes could be rational. Since then, with a few exceptions, almost all work on freeriding and incentives in P2P has been derived from this concept of rationality, and such works have taken the rational action framework for granted. So while noting that behavioral economics demonstrates that purely self-interested models usually fail to explain observed behavior of people [28], Feldman et al. still go on to assume in most of their works "that all individuals are strategic i.e.,they are rational users, who act rationally to maximize their own benefit" [29, 30, 33, 64].

Yu and Singh highlight the fact that most research on peer to peer systems that had been carried out till then (2004), had focused primarily on protocol design and had ignored the rationality of each peer [136]. In a similar vein, Ngan et al. emphasize the point that P2P systems "must be designed to take participant incentives and rationalities into consideration" [86, 87]. In a work where they present a lightweight currency paradigm for the P2P market, Turner and Ross, like others before them, highlight that "unfortunately, peers are typically rational and are thus reluctant to volunteer their resources" [126].

Nandi et al. in aiming to provide incentives in cooperative content distribution systems, assume that users in P2P systems, "resemble economically 'rational' agents who are willing to follow the protocol only if that behavior maximizes the node's utility from the P2P network" [84]. Similarly Xiong and Liu in modeling peer to peer communities, assume that the users of these communities are rational [133].

Schneidman and Parkes consider how to create provably faithful specifications on networks with rational nodes [114]. Nielson et al. assume that "most P2P nodes are rational and will attempt to maximize their consumption of system resources ... if such behavior violates system policy then it constitutes an attack" [88]. Also, many works have presented cheating BitTorrent (the famous file-sharing P2P protocol) clients and conclude that the incentives in BitTorrent are insufficient [12, 93, 116].

2.2 Freeriding and Incentives in P2P: A Taxonomy

We have observed that works done on incentives in P2P systems generally share a common trait, which is the identification with the rational choice model. Cognizant of the fact that this trait operates in the background and implicitly or explicitly guides the choices taken by designers working on incentives, we now proceed to the classification of the works done on incentives in P2P file-sharing systems. Figure 2.1 depicts these classes,

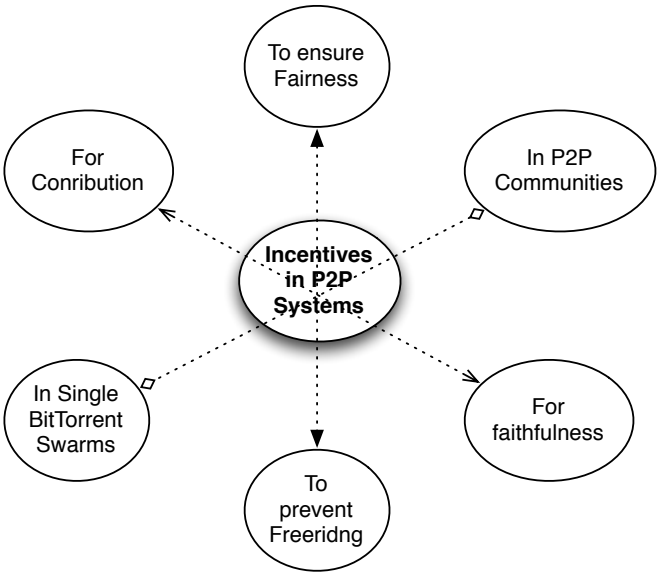


Figure 2.1: The various categories and goals of incentive mechanisms in P2P file sharing systems.

which are not all mutually exclusive and usually overlap.

Single BitTorrent swarm vs Community level.

We can distinguish between works that focus on avoiding freeriding and incentivizing cooperation in a single BitTorrent swarm and those that focus on incentivizing cooperation across P2P communities. The former, of which [11, 66, 70] are examples, assert that the incentives in the original BitTorrent protocol are insufficient. They demonstrate through practical techniques that BitTorrent is flawed in its incentives [72, 93] and usually also propose methods by which modifications in the BitTorrent protocol would ameliorate these shortcomings.

On the other hand, works falling in the latter group, focus on cooperation across entire communities. Most of these papers present some kind of monetary, or indirect reciprocity, mechanisms such as [36, 102, 117].

Incentives for contribution vs Incentives for faithfulness.

The way freeriding and incentive mechanisms have been approached in the literature can be broadly classified into two categories. Some people have approached it like a security issue where clever manipulations and cheats in the protocol are explored. It is shown how such tricks can benefit the cheating client, implicitly demonstrating the lack of incentives for faithfulness in the system, where faithfulness refers to 'following the prescribed protocol'. On the other hand, others have looked at incentives for (more) contributions.

Examples of this approach include works which study the BitTorrent protocol and demonstrate that the protocol generates automatic freeriding in the system [60]. Other examples, not limited to BitTorrent, include works that explore mechanisms to increase contribution levels across the community [80].

Varying definitions of freeriding and fairness.

Underpinning the above approaches are the ways in which freeriding has been defined by different people taking different approaches to the problem. Some people define freeriding as not uploading anything at all [1], some characterize it as uploading less than downloading [6, 11], while some others characterize it as lack of seeding [102, 117]. Directly related to the various ways in which freeriding is studied are the various definitions of "Fairness", a term that is widely used in the literature. Some claim that the BitTorrent protocol is inherently unfair as it does not follow a strict TFT policy and some peers end up contributing much more than others [11].

2.2.1 Overview of some representative works

The motivation behind many "Single BitTorrent swarm" dealing with "fairness" works, is that the incentives in BitTorrent are not fair. Bharambe et al. note that a P2P system should be fair in terms of blocks served by individual nodes. No node should be compelled to upload much more than it has downloaded. Therefore, asymmetries of contribution should not be systematic [11]. The claim is that if there were such asymmetries, the system would not be fair and there would be a lack of incentive for nodes to participate. The paper goes on to show through simulation experiments that BitTorrent is an unfair system, where some nodes end up contributing much more than they download. This happens because of BitTorrent's coarse approximation of TFT, based on limited download estimations. In BitTorrent, faster peers upload to slower peers while getting relatively little in return. This has been deemed to be "unnecessary" in some works. There are several studies such as [35, 39, 60] that argue for a *Bit level tit for tat* fairness. Bit level fairness implies that a fair system should implement byte level reciprocation. So if peer x gives b number of bytes to y, it should get the same or approximately the same (within the range dictated by a given threshold) number of bytes from y in return. The logic is that free riders should be punished and reciprocation should be mandatory.

In [66], it is argued, we think correctly, that Bit level fairness is not appropriate in the context of peer to peer file sharing systems. With Bit level fairness, when there is more capacity of service in the system than request for the capacity, the excess capacity will be lost even if slow leechers or free riders could benefit from it. Bit level fairness does not take into account the fact that peers can (and do in most cases) have asymmetrical network connectivity, the upload capacity being lower than the download capacity. So with Bit level fairness, such peers would never be able to utilize their full download potential.

Also, seeders have no way to ensure Bit level fairness as they do not want anything in return.

Another implication of Bit level fairness or volume based fairness, as it can also be called, is that if faster peers were to interact more with faster peers, rather than wasting resources on slower peers, they would get even faster download times. However it should be noted that conversely, this would mean that the slower peers download times get even slower. At the same time, the freeriding problem would have been solved since slower peers would only be interacting with other slower peers, hence generally consuming as much as they contribute. Work has been done in this direction where reciprocation is tightened to ensure that peers do not 'waste' bandwidth on those others who do not reciprocate at the same level [6, 25, 68].

Despite a very sophisticated modification of the BitTorrent protocol, the results in [68] show that the average download time of peers decreases only slightly as compared to the original BitTorrent protocol. What is more, works such as [65] show that despite not enforcing Bit level fairness, BitTorrent is already good at clustering peers based on their speeds. And, due to BitTorrent's optimistic unchoke policy, slower peers are able to gain benefit from faster peers. However, at the same time BitTorrent is fair in that faster peers, i.e., peers who contribute more are rewarded by completing their download sooner. Perhaps, this is a better criterion of fairness.

However, despite this, as observed, there have been *several* works that aim to modify the BitTorrent protocol in order to make it more "fair". Responding to such arguments, Piatek et al. [93] argue that in BitTorrent, faster peers end up helping slow peers at only a little loss in download speed. If the faster peers help out slower peers, they do it as part of the somewhat egalitarian nature of the protocol. Also, this should not be changed as this is what keeps the system going for a diverse number of peers having wide-ranging bandwidth capacities.

There have also been several interesting "single swarm" works that study "Incentives for faithfulness" in BitTorrent such as [12, 70, 93]. Piatek et al. have done extremely interesting work that intersects many of our identified classes. Firstly, they show that the incentives in BitTorrent are not fair in that fast peers provide low capacity peers with an unfair share of the data. However, they recognize that there is a kind of "progressive tax" on faster peers in BitTorrent, according to which the more and faster that a peer contributes, the better the peer's performance but not in direct proportion. As discussed above, they also argue that building Bit level fairness in BitTorrent would be a failed endeavor because the "majority of the BitTorrent users benefit from this kind of unfairness".

They then turn their attention to "Incentives for faithfulness": they build a strategic and selfish client which can invest its upload capacity in a more efficient manner by not being faithful to the protocol and not sharing its upload bandwidth uniformly with others. Similarly, in [12] a client is presented that is determined to freeride and seeks to complete

a download without uploading anything at all.

We now focus our attention on incentives at the "Community level". This broad category covers: a) general works on reputation mechanisms and trust; b) works focusing on incentives while making use of reputation mechanisms; and c) works that involve elaborate schemes to make the incentive schemes secure, such as the use of supervisory nodes and/or the employment of cryptographic techniques. The influential work, *The eigentrust algorithm for reputation management in P2P networks* [61] belongs to class (a), while *Incentives for combatting freeriding on P2P networks* by Kamvar et al. [61] belongs to class (b) and finally works such as [126], which uses a public key infrastructure that eliminates the need for trusted third parties, and a micropayment scheme based on top of that [135], belong to class (c).

Incentives at the "Community level", or across P2P networks, to put it more accurately, is a bigger area of research as compared to incentive studies specific to the BitTorrent protocol. Works dealing with incentives for cooperation in P2P system can be classified into two categories: *Monetary Payment Schemes* and *Reciprocity Based Schemes*. These classes have also been delineated elsewhere [28]. We discuss them in turn next.

a) Monetary schemes. In *Monetary schemes*, a user's credit (or budget) gets reduced on each instance when it downloads content. Similarly the user's credit increases every time it uploads content. These schemes build upon three things namely: *1) A virtual currency*, *2) Micropayments*, and *3) An accounting structure*.

In a P2P setting there are issues in maintaining this structure. For the accounting structure, such schemes usually have to rely on trusted accounting centers or third parties. Sirivianos et al. present monetary exchanges facilitated by a centralized bank [117]. Great emphasis is laid on creating a non manipulable scheme of exchanges using cryptographic techniques. The presence of a centralized bank means that the scheme is not scalable but has greater security than a completely decentralized solution.

Vishnumurthy et al. present a system involving a virtual currency where sets of bank nodes keep transaction balance of peers [128]. They define *Karma* as the value which captures the amount of resources that a peer has contributed and consumed. This represents the user's standing in the global system. Importantly, the level of Karma (or credit) in the system is maintained, and measures are taken to avoid inflation and deflation that can occur when peers leave the system. In this way, [128] is an important contribution because this work begins to explore the problems that are inherent in dealing with credit systems. In avoiding inflation and deflation, the work's only aim is to maintain the per-capita karma i.e. the total Karma divided by the number of active users. However, willful hoarding of credits, as discussed by Kash et al. [62] is not considered.

Credit crunches and crashes have been studied in Scrip Systems by Kash et al. [62]. They show that in a P2P system, both an overabundance of money supply and its shortage can lead to inefficiency. An overabundance in the money supply leads to a monetary

crash where no one is willing to work and freeriding is encouraged. On the other hand, a shortage in the money supply leads to peers going broke and not being able to afford services in the system.

b) Reciprocity schemes. Now we come to the second category of incentive mechanisms at the "Community level", which are *Reciprocity Schemes*. In *Reciprocity schemes*, users keep a history of past interactions with other peers and use that to inform their decisions. These schemes can either be based on direct reciprocity or indirect reciprocity. BitTorrent is an example of a direct reciprocity scheme. Andrade et al. also present a direct reciprocity mechanism to reduce free riding in P2P CPU sharing grids [3] in which peers compute reputations of others peers based on past interactions. Sun and Molina present a selfish link based incentive mechanism to foster cooperation in P2P systems. Each user keeps statistics about its neighbors and rates them based on how they have behaved towards it in the past. Based on this rating, a node decides how much service to provide to its peers [122].

The authors assert that their incentive structure would incent nodes to either increase the amount of capacity to service neighbor's queries (basically bandwidth) and/or share more data. It goes without saying that such a mechanism rewards peers with high bandwidths. Such peers would naturally have a better reputation at other peers while slower peers even if there were not deliberate free riders would suffer and would not be able to earn higher reputations. Of course, slower peers could earn high reputations by sharing more content. However, to share content, they will first have to acquire it. And they will be much slower than faster peers in acquiring content. This would lead to an ever increasing gap in the reputation of faster and slower peers, making the system highly unproductive for the slower peers. This mechanism resembles the modified BitTorrent protocol presented in [68].

The problem with direct reciprocity schemes is that they depend on private histories. In large P2P networks, spanning millions of users, it is highly unlikely that repeated interactions between the same two peers would take place. Therefore direct reciprocity schemes are not scalable. In order to address this problem, many *indirect reciprocity* schemes have been presented in the literature. These schemes are better known as Reputation Systems.

Scrivener is a decentralized system for ensuring fair bandwidth exchange in cooperative content sharing networks. It enables indirect reciprocity by identifying a credit path from a source node to the node which has the content desired by the source node. In a credit path, each node in this path has credit with the next node [84]. When nodes join the system, they have no credit. To bootstrap a node, the system provides it with some initial content that it can serve to other nodes.

Works such as [36,38] keep track of users contributions in a decentralized manner and use this to determine the reputation of peers. Reputation and Trust mechanisms have been

studied and categorized, in more detail, elsewhere [77, 108].

2.3 Salient Observations

The overview presented in this chapter, allows us to make the following salient observations, which will guide our subsequent work:

1) The economic ideas that have been utilized for incentive related works in P2P systems are imbued in the rational action framework. The assumptions entailed in this framework drive the solutions to free-riding in P2P incentive works, and also impinge upon the usage of terms such as fairness, social welfare, etc. We note that at this juncture we are not arguing for or against the merits of the application of the rational framework for the design of P2P systems. Rationality is an idealization that has been used by economists to study complex situations. In listing the above examples, we only note that in P2P it is mainly the rational framework that has been the source of inspiration from the field of economics. Most works, of which many instances have been described above, have chosen to model peer to peer users as self interested rational agents; and in such a scenario, the goal of the system designers is the welfare of the system when all of its participants are self interested agents. Thus as Tamilmani et al. suggest, *the emphasis of much work in P2P is on achieving a Nash equilibrium when all peers are rational* [123].

In Chapter 6 and 7, we argue for the consideration of alternate viewpoints viz a viz concepts such as fairness, freeriding and social welfare; and for formulation of alternatives to game-theoretic solution concepts for assessing the robustness and performance of distributed protocols.

2) We observed that Kash et al. showed that credit crunches and crashes can occur in systems which utilize a virtual currency. Most private BitTorrent communities impose sharing ratio enforcements or a credit based scheme for incentivizing peers' contribution. Such communities maintain centralized accounts for peers and record their upload/download behavior. Peers that do not maintain a good ratio over some time period are prevented from downloading more content till they build up a better ratio by uploading content. Studies such as [16, 71, 137] have shown that such sharing ratio enforcement in these private BitTorrent communities does induce people to seed more. However, we indicate in Chapter 4 that such communities can also face problems such as credit crunches. In Chapter 5, we show that such problems are exacerbated when peers do not follow the rational user model. We also present some ways in which these problems can be rectified.

Chapter 3

Robust vote sampling in a P2P media distribution system

As discussed in Chapter 1, the BitTorrent protocol has transformed the distribution of large media files due to its decentralized, scalable, efficient, and robust peer-to-peer (P2P) architecture [18]. Peers share bandwidth to help distribute files of common interest. However, the protocol excludes mechanisms for searching, rating, and associating descriptive metadata to content.

Consequently, to locate high quality content available for download, users often rely on web-based systems that provide both links to content (.torrent files) and associated metadata such as a text description, a thumbnail graphic, a URL to associated information on the web, and other useful information. This allows users to search and browse available content before making a selection of what to download.

Web-based systems rely on user contributions of both .torrent files, that point to available content, and metadata that describes content. They are administered centrally so malicious content or incorrect metadata can be manually removed and the users who posted them excluded. Creating new identities in such systems involves some level of user cost since identities must be created on the web system prior to posting information. Hence such systems cannot entirely stop anti-social behavior, but they make it costly because creating new user identities involves some time and effort.

A number of BitTorrent clients have, recently, integrated their own search and metadata systems such that users can locate and browse available content conveniently from within the client before downloading (e.g. Vuze[1] and Miro[2]). However, again, these rely on centrally administered servers to store and serve metadata requiring the creation of user accounts.

In this chapter we propose a design for a fully distributed metadata dissemination and

[1] http://vuze.com

[2] http://www.getmiro.com/

rating system which provides similar functionality to centralized systems. Low quality metadata such as spam or incorrect information is combated through a distributed ranking system based on the sampling of user votes.

Our approach is a major step towards a completely decentralized, and self-maintaining, BitTorrent media sharing community[3].

As with centralized systems we do not eliminate the possibility of anti-social behavior but we make it costly and difficult without the need for central servers or administration.

The target platform for our design is the Tribler[4] media client [96] but the design is generic enough that it should be applicable in other media sharing contexts where decentralized and robust metadata dissemination and rating are required. Tribler has a non-spoofable distributed peer identity system using a public key infrastructure. This means that all communication between nodes is signed and bound to a known source identity, thus preventing forged or stolen identities. Also, the Tribler client provides local database services allowing state to be maintained over sessions.

The chapter is structured as follows. In Section 3.1 we give an overview of the system. In Section 3.2 we briefly discuss the peer sampling service (PSS) - which provides an essential service required by our design. In Section 3.3 we describe our metadata dissemination mechanism. In Section 3.4 we present our vote sampling approach and in Section 3.5 we present simulation results based on real BitTorrent peer traces. In Section 3.6 we discuss some vulnerabilities and possible refinements of our design. In Section 3.7 we present a summary of related work. Finally we conclude with a summary of our contribution and possible future work.

3.1 Design

In this section we describe, in outline, the overall design of our system giving the reasons for our main choices.

We require a method by which nodes in a P2P file sharing system can submit, distribute, and rank metadata while respecting the constraints of: 1) Full decentralization requiring no servers or centralized components; 2) Scalability to millions of nodes; 3) Resistance to malicious attacks which attempt to display spam metadata to the user.

In order to propagate and store metadata we selected a gossip (or epidemic) based replication approach. Each peer stores metadata in its own local database. By storing metadata locally we ensure that it has high availability. Periodically peers are paired randomly and exchange metadata updating their own local databases (Section 3.2 discusses the peer sampling service (PSS) mechanism used). We selected a gossip based design

[3] Which would also require the use of a decentralized tracker. This is an active area of research and there are also deployed versions in use but this is beyond the scope of this chapter.

[4] http://tribler.org

because it requires no central components and is robust to high churn rates [22]. We could have stored metadata in a Distributed Hash Table but these require explicit leave and join operations which are costly in systems with high churn, such as file sharing networks [120]. Additionally, search performance is considerably enhanced if metadata is stored locally because it is not necessary to perform multi-hop look-ups.

We require that users can "vote" on metadata indicating if they consider it to be of high quality (a positive vote) or spam (a negative vote) and that these votes can be communicated to other nodes such that they can be used for ranking items after a user search. Here we made two design decisions influenced by security and efficiency concerns.

Firstly we decided to bind votes not to metadata items (which we term "moderations") but to the users who created them (which we term "moderators"). Our thinking here is that moderators will either tend to be good or bad - i.e. to create quality moderations or spam moderations. Also this makes more efficient use of user input. Since it is known that users rarely vote or moderate items in file sharing networks [5]. We cannot expect each moderation to obtain sufficient votes to rank them. However, given a small number of active moderators and a large number of moderations, the few votes made by users, when bound to moderators, can produce sufficient quantity for meaningful ranking.

Secondly we decided, for security reasons, to only count votes from nodes encountered directly via the PSS. Hence we sample the population randomly rather than aggregating votes using gossip based aggregation methods [58]. This ensures that each node can only vote once for any moderator (a one node one vote per moderator policy). The downside of this decision is that each node requires sufficient time to obtain a good sample and also that different nodes will have different samples at any given time. Hence we trade speed and efficiency for security.

In order to frustrate spam moderators we utilize a distributed mechanism based on an *experience function* which imposes a cost on new identities before their votes are accepted by other nodes. However, this cost would be incurred anyway by a normal functioning node in a file sharing network (uploading files to others). Hence we do not impose an artificial cost that would degrade the efficiency of the system. The experience function is presented in Section 3.4.2. Here we trade some extra protocol complexity for security against spam attacks where many cheap identities are used to vote up a spam moderation.

3.2 Peer sampling service

We assume each peer has access to a peer sampling service (PSS) which periodically returns a random peer from the entire population of online peers. This allows nodes to discover others and potentially exchange messages with them.

[5] Based on data from YouTube and a popular BitTorrent public tracker community, mininova.org, we found typically no more than five user votes per 1000 views or downloads.

There are several ways to implement a PSS in a distributed and robust way. One approach uses *gossiping* or *epidemic protocols*. Such approaches maintain a random-like overlay network in which nodes regularly exchange their neighbor lists (or view) with others. Such PSS protocols have been shown to be robust, self-repairing, completely decentralized, and scalable to tens of millions of nodes [59].

Our target system, the already deployed Tribler system, implements a variant of Newscast [129] called BuddyCast [96].

3.3 Metadata dissemination

In this section, we will describe how metadata is disseminated in our system. The low level details of the metadata dissemination protocol, called ModerationCast, and extensive simulations, are given in [46]. Here we present the main features relevant to the voting mechanisms described later.

Moderations are disseminated in a gossip-like fashion to other peers by using the PSS. However, nodes only pass on metadata from those moderators they have approved. Approval involves the user explicitly selecting a thumbs-up icon displayed next to the metadata from the given moderator indicating a positive (+) vote for the moderator. Users may also disapprove of a moderator by selecting a thumbs-down indicating a negative (-) vote.

Over time as nodes encounter others, through the PSS, they will receive new moderations either directly from the moderator, if they encounter them, or from those nodes which have approved the moderator. Received moderations are stored in a local database. Hence highly approved moderators will tend to spread their metadata more quickly than moderators that are not highly approved. If no other node approves a moderator then the only way that its metadata can spread is through direct contact with other nodes. Nodes that disapprove a moderator remove all associated moderations from their local database and refuse any new moderations from that moderator.

Essentially then, the idea is that, "good" moderators, as judged by the approval of others, will spread their metadata quickly but "bad" moderators, obtaining low numbers of approvals and / or disapprovals, will only be able to spread their metadata slowly. However, it is important to note that even bad moderators can spread their data to others through direct interactions with nodes that have not already indicated disapproval. Figure 3.1 shows a schematic diagram showing how a moderation spreads in the population based on approvals and disapprovals by other nodes. The figure shows a moderator node that creates the metadata (m) and nodes that give the moderator positive votes (+). Shaded nodes have received and stored the metadata in their local_db. Nodes that give negative votes (-) or no votes (null) do not pass on the metadata.

To authenticate moderations we use digital signatures. This prevents alteration of

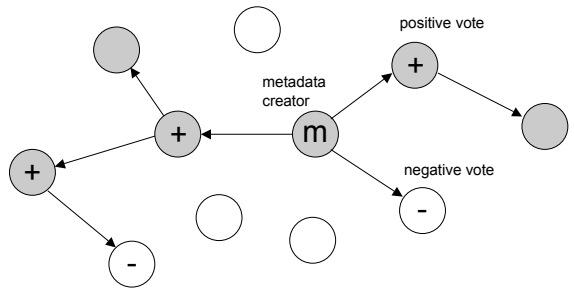


Figure 3.1: Diagram illustrating how moderations are spread.

```
do forever
  wait Δ
  j ← GetRandomNode()
  ml_i ← Extract(local_db)
  Send ml_i to j
  ml_j ← Receive(j)
  local_db ← Merge(local_db, ml_j)
```

(a) active thread

```
do forever
  ml_i ← receive(*)
  ml_j ← Extract(local_db)
  Send ml_j to i
  local_db ← Merge(local_db, ml_i)
```

(b) passive thread

Figure 3.2: The push / pull gossip based metadata dissemination protocol.

moderations by malicious peers. Figure 3.2 shows outline pseudocode for the gossip based metadata dissemination protocol. GetRandomNode is supplied by the PSS. The local_db stores received moderations. The Extract() function returns the moderations list (ml) sent to other nodes. The Merge() function inserts new moderations into the local_db. These operations take account of local node votes and moderation recency criteria.

3.4 Vote sampling

In this section we describe how votes are sampled and collected by peers in the network. We designed two related protocols: BallotBox and VoxPopuli and an associated *experience function*.

These protocols support a two-tier identity system because the population is partitioned into an experienced core and an inexperienced periphery. Figure 3.4 illustrates this view of the system schematically.

Figure 3.3 shows outline pseudocode for the two protocols. We describe each in turn below.

3.4.1 BallotBox protocol

As previously stated votes are generated by users registering approval (a positive vote) or disapproval (a negative vote) against moderators. Each peer node stores a list of the votes

the local user has made in a structure we term the *local vote list*. Each entry in the local vote list contains a pair mapping of a unique moderator ID to a vote (either positive or negative) plus a time stamp indicating when the user made the vote. Moderators may only appear once in the list since a user is only allowed to make one vote against each unique moderator. The length of the list therefore indicates the total number of votes cast by the local user. It is a record of their local voting pattern. It can be thought of as a ballot paper that the users fill in as they make votes.

Periodically each node i selects another node j from the population randomly (using the PSS). Each node then applies an experience function E to the other to determine if to request the local vote list of the other (Section 3.4.2 below discusses the E function in detail). Requested nodes communicate their local vote list to the other. Nodes send a maximum of 50 votes, selecting them based on a recency and random policy. Experiments demonstrated that combining these policies produced acceptable performance [46]. Nodes then merge received vote lists into a structure we term the *local ballot box*. The local ballot box is a list in which each entry contains four items: mapping a unique moderator ID to a vote, a time stamp and a unique peer ID. The local ballot box is similar in format to the local vote list except that moderators may appear several times in the list, recording votes for the same moderator received from different peers. Also the time stamp records when the vote was received, by the local node, rather than made by the remote node.

Essentially then, each peer individually conducts its own poll by asking other randomly selected peers directly to supply their local vote list. Hence pairs of peers meet randomly and exchange votes, building, over time, a sample of the votes of the population in their local ballot boxes.

Nodes do not forward or share the accumulated information in their local ballot box with other peers. This precludes certain kinds of malicious vote manipulation where a node could lie about the votes received from others. But this means that each peer can only accumulate a sample of the population votes, based on its direct experience, not a globally accurate total count. Faster and more accurate epidemic-style aggregation protocols have been proposed but they are highly vulnerable to lying behaviour [58].

The local ballot box has a maximum size of B_{max} votes from unique peers - beyond which new votes replace the oldest votes. Hence BallotBox determines voting statistics from a maximum sample of B_{max} other peers. Assuming the PSS produces random samples and B_{max} is large enough then we can expect the local cache to converge to a reasonable accuracy.

The BallotBox protocol, therefore, is similar to an “opinion poll” as carried out by polling organizations when attempting to determine the opinion of an entire population on some matter of interest. In general such polls ask individuals directly their own opinion but not what they believe others opinions are. The BallotBox turns every peer into a

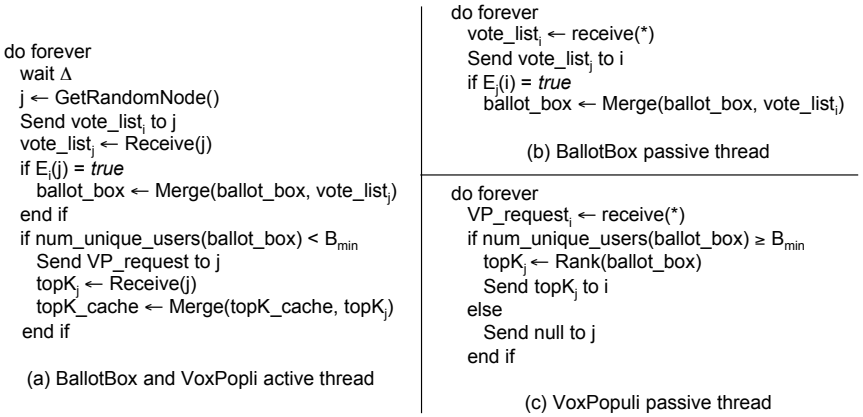


Figure 3.3: The BallotBox and Voxpuli protocols.

pollster enquiring on moderators.

Based on the current contents of the local ballot box a peer can calculate a ranking of moderators. This, in turn, can be used to rank metadata items to which the moderators are bound. In order to take the raw votes from the local ballot box and produce a ranking of moderators any suitable method could be applied such as simple summation or more complex proportional approaches. We do not discuss this further here.

Another possible use for the vote sample information is to display a screen listing the top-K moderators themselves along with their estimated percentage of the popular vote and other associated information. We believe such a screen could psychologically incentivize moderators to produce good moderations since they can see themselves rise in the ranks of listed moderators as others vote for them.

3.4.2 Experience function

Since new identities are cheap in our system, immediate voting power would enable Sybil [23] and collusive flash crowd attacks (where a large number of new peers join the system with the explicit aim of promoting a moderator by voting for them for nefarious purposes). We therefore enforce that new nodes joining the system cannot register votes with others until they are considered "experienced" by the receiving nodes. In other words, any node should only take a vote from another node into account when this other node is experienced.

We define the general *experience function* as a binary function E that determines whether or not a node is considered to be experienced. Any candidate E function must be implementable in a fully distributed way and be robust to attempts to fake experience (i.e. to immediately and cheaply generate experienced identities). While the overall design of our voting system does not assume a particular definition of experience, in the

implementation that we will present in this chapter we consider a node experienced when it has contributed a certain amount of data to the peer population by sharing files.

In order to determine the contribution of a node in a safe way we use the BarterCast protocol [80], which is deployed in the Tribler system. In essence, by using BarterCast, any node in the system can estimate the contribution of any other node (that it knows via the PSS) based on up- and download statistics that are exchanged among nodes in a reliable way. First, nodes record statistics of their own BitTorrent file-transfers. Second, nodes exchange their own direct statistics with other peers they encounter. Based on these combined statistics each peer can build a graph of the network with directed edges that denote the amount of MBs transferred from one node to another node. The protocol then applies a maxflow algorithm to derive peer contributions. The maxflow approach highly reduces the effect of (collaborations of) nodes that provide false information in order to fake experience. This approach and its properties are described in detail in [31] and [80].

In the remainder of this chapter, we assume that node i can compute a reliable *contribution value* for a node j. We denote the contribution of j to i by $f_{i \to j}$. For the purposes of implementing our experience function E, we apply a simple threshold value T over the contribution function $f_{j \to i}$. Hence node i considers node j to be experienced where:

$$E_i(j) = \begin{cases} true & \text{iff } f_{j \to i} \geq T; \\ false & \text{otherwise.} \end{cases}$$

In our current design we determined an appropriate T value via simulations based on real BitTorrent traces (see Section 3.5). However, as we discuss later, T could be adapted dynamically to adjust to differing conditions.

3.4.3 VoxPopuli protocol

The BallotBox protocol involves nodes constructing a sample of votes based on direct interaction with individual peers, who are considered experienced. Therefore, new nodes entering the system need time to build a reasonably sized sample before they can extract meaningful statistics. A minimum sample size of B_{min} votes from other peers is required before any statistics are used. This is to avoid artefacts produced by extremely low vote samples. During this bootstapping process nodes use the VoxPopuli protocol - a less accurate but speedier protocol which dispenses with the need for individual vote counting or the application of an experience function.

When executing the VoxPopuli, nodes request from others (encountered via the PSS) a rank list of the top-K moderators. No experience function is applied. Only those nodes which are *not* themselves executing VoxPopuli respond based on their current local ballot box statistics - by producing a ranked list of moderators truncated to a maximum size of K and sending this to the requesting node.

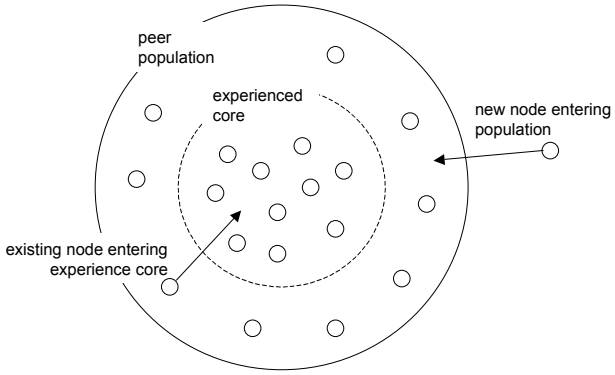


Figure 3.4: Diagram illustrating the concept of a peer population comprising an experienced core of nodes and new nodes entering the system. Over time new nodes will enter the experienced core as other nodes recognize them as experienced.

Each node executing VoxPopuli maintains a local cache of the last V_{max} top-K lists received and performs a merge operation to produce its own top-K list which is used for relevance ranking. There is no minimum sample size required so after the first VoxPopuli exchange a node can rank moderators.

Any rank merging method could be used. We apply simple averaging of the rank of each moderator over all stored top-K lists. Where a moderator does not appear in a list they are assumed to have rank K+1 for that list.

VoxPopuli is therefore quick (in the order of a few seconds) but vulnerable to inexperienced and colluding malicious peers - if a sufficient number enter the system such that they form a large enough proportion of the total peer population and they promote the same spam moderators.

3.5 Simulation results

In order to test our proposed design we performed simulations based on real traces from the private BitTorrent tracker *filelist.org*. It is possible to trace the behavior of unique peers over multiple sessions and multiple swarms with this tracker. This is not possible with public trackers such as *mininova.org*. The traces record the size of the files that are shared in each swarm, and uptimes, downtimes and the connectability of peers (i.e. if they are behind a firewall or freely connectable).

The traces capture the realistic high churn rates found in deployed P2P systems. On average only 50% of the total population of nodes are online at any given time[6]. For our simulations we used a dataset of 10 unique traces of 7 day duration monitoring 100

[6]By total population we mean all nodes that enter the system, over the seven day period of the traces.

unique peers. Each trace records approximately 23,000 unique events making a total of 2.3×10^4 events[7].

Our simulations operate at the BitTorrent file piece level. This means we simulate every action that a BitTorrent client would need to take, down to the exchange of file chunks, peer choking and piece selection. We simulate nodes that are predisposed to seed files altruistically after they have downloaded and those which free-ride by leaving swarms as soon as they have downloaded their file. Such detailed simulations are non-trivial to implement and require significant computational power to run.

In the following sections we perform simulation experiments that demonstrate: The generation of an experienced core - Section 3.5.1; The performance of vote sampling protocols - Section 3.5.2; The performance of the system under spam attacks by a massive number of colluders who attempt to promote spam metadata - Section 3.5.3.

3.5.1 Experience formation

We used trace based simulations to determine how quickly our system would produce an experienced core for given threshold values T for the experience function E (as described in Section 3.4.2). In order to gain a general picture of the formation of experience between nodes we calculated a time series for a value we term the Collective Experience Value (CEV). Since E is a binary and non-symmetrical function that can be applied to any pair of ordered nodes we calculated the average of E applied to all ordered pairs of nodes i, j where $E_i(j) = true$ contributes a value of 1 and $E_i(j) = false$ contributes a value of zero, hence:

$$e_i(j) = \begin{cases} 1 & \text{iff } E_i(j) = true, \\ 0 & \text{otherwise;} \end{cases}$$

and

$$CEV = \frac{1}{N} \sum_{i \in N} \sum_{j \neq i} \frac{e_i(j)}{N-1},$$

where N is the known total population size[8].

The CEV is therefore a form of directed graph density value in which edges are defined between nodes based on the experience function E. It characterizes the amount of experience between nodes. Figure 3.5 shows simulation results based on a typical trace from the dataset for various threshold values T. We conducted these series of experiments

[7]The datasets used, and associated documentation, are publicly available at: http://davidhales.com/tom-data.zip.

[8]The CEV value is therefore a measurement requiring global information and cannot be calculated by individual nodes easily. We use our global knowledge of the traces to calculate the CEV value for our own experimental purposes. CEV plays no part in the protocols running in the nodes.

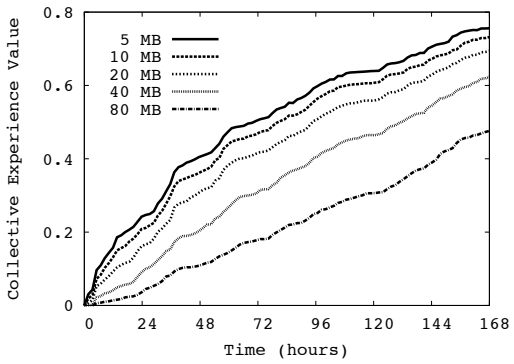


Figure 3.5: Chart showing the Collective Experience Value (CEV) over time for a typical trace of BitTorrent activity.

in order to ascertain the time required for the generation of a sizeable experienced core for different T values. In the figure, the CEV indicates the proportion of all ordered pairs of nodes (i, j) for which i considers j to be experienced. Hence when CEV=1 all nodes consider all others to be experienced. Results using five different threshold (T) values for the experience function E are shown.

Based on the results shown in Figure 3.5 we selected the lower value of $T = 5$ MB for use in the further simulation experiments discussed below. This value was chosen pragmatically. The results indicate that approximately 20% of ordered node pairs produce experience within 12 hours. This indicates the early formation of a core but implies a time and upload cost is necessary before a node can enter it. This frustrates Sybil or flash crowd attacks while placing only modest costs on honest new peers entering the system.

Note, however, that even after seven days (168 hours) some nodes are still not part of the core. This is due to freeriding behaviour in which nodes download but upload little[9] and the fact that some nodes are rarely present in the trace - they may enter and quickly leave the system spending most of the time offline. As stated previously, analysis of the traces indicated that churn rates are high with approximately 50% of nodes offline at any given time. This indicates the traces reflect actual BitTorrent client file sharing dynamics. It is important to note that the results in Figure 3.5 show results for all peer nodes that comprise the total population over the seven day period not just those who are online at any given time. It therefore gives a global picture of the traced population rather than online snapshots.

[9] Analysis of the traces showed that approximately 25% of peers uploaded little to others.

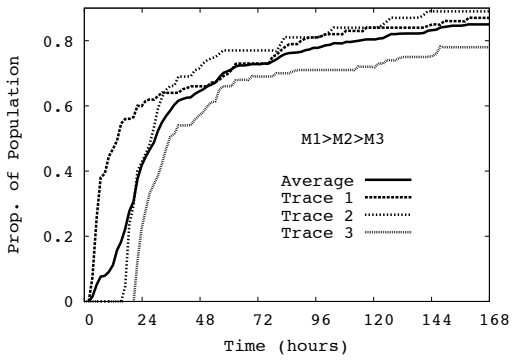


Figure 3.6: Chart showing the effectiveness of the vote sampling system over time.

3.5.2 Vote sampling

We tested the vote sampling protocols by running simulations in which a small number of moderators spread metadata and are allocated votes from other nodes when they receive this metadata. We set the first three nodes (M1, M2, and M3) entering the system to be moderators and to spread a moderation related to a .torrent file. We selected 10% of the population at random to provide a positive vote for M1 and 10% to provide a negative vote for M3. M2 gets no votes. Hence the correct ordering, based on the popular vote, should be $M1 > M2 > M3$.

For the BallotBox we set $B_{min} = 5$ and $B_{max} = 100$. The VoxPopuli cache size was set to $V_{max} = 10$ and for the top-K moderator lists $K = 3$.

The results for a typical run are shown in Figure 3.6. The correct ordering is $M1 > M2 > M3$ based on votes. Voting nodes do not vote until they receive the appropriate moderations via the moderation dissemination protocol. Three typical runs are shown for different independent traces. The average is over 10 independent runs. Notice that at approximately 12 hours there is a sharp rise. This is result of the bootstrapping properties of the VoxPopuli protocol. Recall that VoxPopuli allows nodes which have not yet received at least B_{min} votes to determine an ordering of moderators by randomly sampling the population requesting the top-K moderators from others who have received at least B_{min} votes. The steep rise indicates that a number of nodes have obtained these thresholds and can thus spread the top-K moderators to others. Hence nodes who have attained the required BallotBox sample size, since they have received enough votes from core nodes, share their rankings with new nodes entering the system - who cannot distinguish core nodes from other new nodes.

3.5.3 Spam attack

A major threat to our system is an attack on new normal nodes - those which have not yet received at least B_{min} votes from core nodes but act honestly - by a flash crowd of new nodes promoting a spam moderator. Such a flash crowd could be comprised of colluding nodes or the result of a Sybil [23] attack in which a single node creates and maintains multiple identities. In either case it is clear that if the size of the spam crowd is significantly larger than the online experienced core then the spam attack will be successful against newly entering normal nodes. However, over time, new nodes will recover from the attack when they have obtained at least $B_{min} = 5$ votes in their local ballot box. So correctly functioning new nodes will only be vulnerable to such an attack for some initial period of time. Figure 3.7 illustrates this kind of attack diagrammatically. Dotted arrows show the flow of protocol messages between subpopulations: VoxPopuli (VP) and Ballot-Box (BB). Notice that the flash crowd cannot influence the experienced core because core nodes do not considered new nodes to be experienced. Over time newly entered normal nodes will eventually become experienced and enter the core.

Figure 3.8 gives results from simulations based on the 10 traces. However, here we fixed 30 nodes to be part of the experienced core. We set the flash crowd size to 30 and 60 in size. The population comprises an experienced core, a collusive flash crowd plus other newly arrived normal nodes. At the start of the run the entire core is converged on a top moderator M1. The flash crowd promote M0 as top moderator (representing a spam moderator). Lines show the proportion of newly arrived nodes ranking M0 top for different sizes of the flash crowd relative to the core size. For attack sizes smaller than 1 x core size, zero pollution is obtain within the first hour. Averages of 10 trace runs are shown. As can be seen when the flash crowd is 2 x core size (60) then most new nodes are defeated (polluted by a spam moderator - M0) for approximately 24 hours. When the flash crowed is 1 x core size then only a minority of nodes are defeated initially.

3.6 Discussion

The claim that the experience mechanism can discourage spam rests on two assumptions: 1) that the experienced core, over time, will become large and be populated by non-spam (or non-spam voting) nodes and 2) that spam nodes will not be able or prepared to pay the upload cost to join the experienced core.

Our first assumption is based on the idea that a population only becomes desirable for spamming if it is already large and that any small population that is already infected by spam nodes will not become large since users will go elsewhere and join other more productive communities. Hence we believe it is not unrealistic to assume that the founders (or elders) of any successful peer community are unlikely to be malicious. To put this

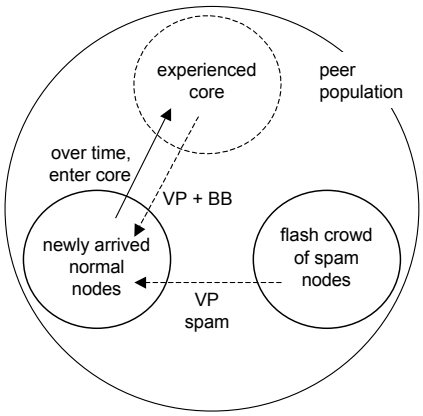


Figure 3.7: Diagram illustrating a flash crowd attack on a population comprising an experienced core plus other newly arrived nodes.

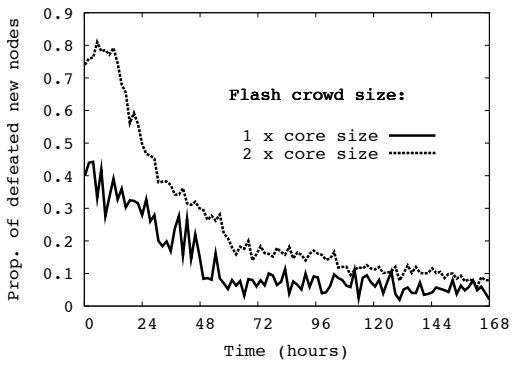


Figure 3.8: Results of a spam attack.

another way, no successful community can form if the founders are malicious. Hales and Patarin have made a similar point to explain why people do not cheat the BitTorrent protocol [42].

Our second assumption relies on the fact that to gain enough experienced identities to influence the popular vote the spam nodes would need to pay a high price in time and upload bandwidth - a price that would be too high to make an attack worthwhile. The larger the size of the core, the higher the cost of an attack since more spam identities are needed to influence the vote. As we have demonstrated, even if the number of new malicious identities are equal in size to the existing core their voting influence is limited. Here we benefit from scaling - the larger the core the better the defence against attack. However, if a sufficiently large set of colluding identities are willing to pay the cost, then our system can be defeated.

We selected a T value using global information and traces of past activity. A more robust approach would be to adapt the T value endogenously based on experience. One possible approach involves nodes starting with a threshold $T = 0$ and adapting it based on the incoming votes. It is reasonable to suppose that honest peers would have the same opinion on the reliability of moderators. We could choose a maximum dispersion level of opinion in votes, D_{max}, above which we increase T. If incoming votes result in an increase in the dispersion level and take it above D_{max}, the value of T is increased and vice versa. The basic idea is that nodes respond to what they deem to be malicious attacks because dispersion in voting may indicate the presence of malicious peers in the system. By increasing T, peers would look to shield themselves from the votes of newcomers and place their trust in more experienced members of the tribe or community.

Another potential flaw in our approach is that *it is possible to fake experience* by clever collusion within the BarterCast protocol but this is difficult and again costly [80]. This is a variant of the so-called "front peer" or "mole" attack [32]

3.7 Related Work

In the literature, the work that bears greatest resemblance to ours in its goals is Credence presented by Walsh et al. [130]. Credence is deployed on top of the Gnutella file sharing network. No metadata dissemination system is present in Credence, instead it relies on the underlying pull based mechanism of the Gnutella protocol for content search. Rather than voting on moderators, peers vote on files in the system. A peer X can evaluate another peer Y's votes based on the correlation in the voting histories of the two peers. Further, apart from simple pairwise matching, peers can leverage the correlations discovered by other peers and compute their own correlations with distant peers by executing a flow-based algorithm. Using this approach, users who do not vote, or do so only minimally, have no way of distinguishing between honest and malicious voters. This is evident from the results presented in [130] where nearly fifty percent of clients are isolated and have no correlations with other peers. In contrast our system does not rely on a large number of people voting, yet still works for all peers, regardless of their voting habits.

System models, analysis, and performance evaluation of unstructured self-managing rating systems (UMR) and structured supervising rating (SSR) systems are presented by Tian et al. [124]. UMR is implemented in unstructured networks where every peer stores its rating of other peers on its superpeer. While, in SSR, each peer has a supervisor peer which is responsible for storing other peers' rating of that particular peer. Each supervisor node is located using a DHT [120] based algorithm.

Our approach of data dissemination and storage is based on a push-pull gossip approach which differs from both these approaches. Furthermore, like most work on ratings, their approach assumes that all peers will issue ratings after each transaction with

other peers. Additionally, they give a detailed account of a "personalized credibility subsystem"' which bears some resemblance to our experience function. Each peer, *i*, maintains a personalized credibility set that contains the peers who vote honestly on *i*. However, unlike our system, ratings from outside the credibility set are also accepted albeit with lower weights. This approach also differs from ours in how it deals with new peers. While we bootstrap the new peers with randomly sampled top-K ratings (see Section 3.4.3), [124] presents no method for bootstrapping new peers.

Kamvar et al. present EigenTrust [61] that uses an algorithm which is similar to the PageRank algorithm by Google [91]. It provides globally consistent trust values for peers in a network using DHT techniques. However, for convergence, it depends on pre-trusted peers and assumes a stable network of peers thus making it unsuitable to networks with high churn. Damiani et al. present a system in which peers ask for ratings on particular peers by broadcasting poll messages [20]. This is basically a UMR system. Xiong et al. present the PeerTrust system [134], in which ratings on a peer are maintained by its supervising peer. This makes it an SSR system in the terminology of [124]. Such "supervisory" systems, due to high churn rates and the unreliable nature of real world systems, are often impractical and not suitable for real world implementation and deployment in open file sharing systems.

3.8 Conclusion

We have presented the design and simulations of a metadata dissemination and ranking system applicable to the deployed Tribler P2P media client. Three protocols were introduced: ModerationCast, BallotBox and VoxPopuli. They work together to spread and rate metadata robustly.

We aimed to produce a light-weight, self-organising, and robust design that is realistic enough to be deployed and requires zero server support. Our work is a stepping stone towards a fully distributed and self-maintaining BitTorrent community that does not require the use of central moderators and servers.

We have built on existing deployed protocols BuddyCast and BarterCast to provide a peer sampling service (PSS) and a robust experience function.

However, we consider our design to be generic enough that other PSS protocols and experience functions could be used. Specifically any distributed trust metric that can rate nodes in a robust way could be adapted as basis for an experience function and hence can be used to support a robust distributed vote sampling model.

We have discussed the vulnerabilities in our approach and indicated why we consider these to be tolerable in realistic environments. We also indicated some future lines of research that may address some open issues. We aim to work on the adaptive threshold T, in order to make our design more robust against attacks.

To our knowledge Tribler is currently the only deployed P2P file sharing system using a fully distributed PSS to augment the BitTorrent protocol to provide proactive media discovery and seeding incentives [80]. To this we add a fully distributed metadata dissemination and rating system.

As of writing, we have integrated our code into the Tribler codebase.

Chapter 4

The Big Crunch in BitTorrent

Originally the BitTorrent (BT) protocol was envisaged as an open protocol in which any peer could participate to cooperatively download files. To share a file a user needs to create a small .torrent file which uniquely identifies the file and binds it to a BT Tracker - a centralized server that keeps track of all the peers interested in a particular file. The .torrent file can then be distributed by any means, such as placing it on a webserver or e-mailing to a list. A user interested in the file can download the .torrent and activate their BT client software. The BT client will contact the tracker and connect to other peers interested in the file. The peers then cooperate to download the file by sharing pieces between them. A set of peers sharing a particular file is termed a "swarm".

As discussed previously, a key feature of the BT protocol is the use of a "Tit-For-Tat" (TFT) strategy to control freeriding [7]. Put crudely, the idea of TFT is to incentivize peers to upload as well as download within a single swarm. Essentially, BT clients will stop uploading (sharing content) with those others who do not reciprocate.

However, although TFT provides incentives for those downloading from a given swarm to upload (to get more download) it does not provide incentives for two crucial activities: 1) for a peer to continue to share the file after it has downloaded the entire file - this is termed "seeding"; 2) for a peer to share a file in the first instance. In addition, if a peer uploads more than it downloads in a swarm it cannot carry over any "credit" to a new swarm.

Recently, there has been a growth of "private tracker" based methods that attempt to provide incentives for these key functions by maintaining centralized accounts, that record upload / download behaviour of peers and apportion "credit" scores, and shutting out users who do not provide reasonable ratio to the system over some time period. We describe this approach in Section 4.1.

We examine statistics from a popular private tracker, TV Torrents, used to share TV shows in Section 4.2. It is evident that such approaches *do* appear to provide incentives

for seeding since we find that there are a huge number of seeders within the system[1]. However, we find that a small minority of peers obtain a large amount of credit in the system. Similar results have also been obtained for another private BT community [4]. Such peers, who hog the majority of the credits in the system, have been defined as "hoarders" in the literature [62]. Hoarders have a negative impact on the system because their presence effectively means that there is less credit in circulation leading to a credit squeeze. As a result other peers suffer, unable to obtain desired services due to being short on credit. This creates a credit squeeze. *We define a credit squeeze as a situation in which, due to lack of credit, the efficiency of the system is significantly reduced.*

Using a simplified simulation model we show that even when all major factors are equal, a skewed distribution of credit in the system emerges and leads to a credit squeeze reducing system efficiency. Interestingly, we show that *adding capacity to the system can, counter-intuitively, reduce performance* due to shortage of credit. We also examine a simple method used by the measured private tracker that ameliorates the credit squeeze in our model. The BitCrunch model description, simulation results and discussion are given in Section 4.3, 4.4, and 4.5.

This chapter reports initial results in what we believe is an under explored research area. Our aim is to open a new line of work, within BitTorrent studies, which examines the role and function of macro-economic policies at community level in order to increase community efficiency.

4.1 Private trackers

Private trackers require users to register before they are allowed to download .torrent files and participate in swarms.

A major function of private trackers is to implement "ratio enforcement". This means the tracker monitors how much upload and download bandwidth is used by each peer over time. If a peer downloads more than it uploads (a form of freeriding) then the download service is terminated. Ratio is persistent over peer sessions and is stored centrally with the tracker. This provides a mechanism for peers to build-up a positive "ratio" or earn credit in the system over many sessions and swarms. This means that peers who seed content to others earn credit that they can "cash in" at a future time and / or in different swarms.

The tracker acts as an accounting system in which the balance is constant over time. When peer i uploads to peer j then i's account is credited and j's account is debited by the amount exchanged (some number of megabytes say). Only those peers with positive credit may continue to download from others.

[1] Having said that, it is quite possible that people in such communities are keen to contribute due to communal solidarity, and not due to the incentive mechanism.

If we assume that the tracker performs ratio enforcement by terminating any downloading by peers with zero credit then the total credit in the community will always be a constant value C but its distribution among peers will change over time to reflect the actual exchanges that have occurred. Hence a peer that seeds a very popular file to many others and has a high physical upload rate would expect to accumulate a high proportion of the available credit whereas those that have downloaded much but uploaded little will expect to have small or zero share of the credit.

Different private trackers run variants of this form of policy. They may allow negative credit or new artificial credit for new. Also, as would be seen, some trackers allow new credit to be created in the system overtime.

4.2 Evidence of a credit squeeze?

Given peers with different file preferences and upload / download rates, the result of such a process often leads to a highly skewed distribution of credit over all peers. In fact, we have found that a small "rich club" of peers appear to hold a large proportion of credit in a real private tracker community.

Statistics from a private tracker

Table 4.1 shows 7 days of statistics gathered from a popular private tracker, TV Torrents (over the period 06/02/09 to 12/02/09). This tracker has a reported daily population of approximately 50,000 peers serving of the order of 10,000 .torrents. Peers earn credit by seeding and spend it by leeching (downloading). It is possible for peers to earn credit by uploading while downloading, through the BitTorrent TFT process, but we found that there is strong evidence that most swarms are over-seeded and hence we discount this as a major factor in credit dynamics of the tracker we measured. In over-seeded swarms downloaders do not have to trade with downloaders but can download directly from seeders.

The tracker we measured rewards seeding by providing credit to seeders at a "bonus rate" of 1.5 times their upload contribution. Hence a peer uploading one byte will receive 1.5 credits while downloading one byte will cost 1 credit. This means the amount of credit in the system is always increasing.

Since we have access to statistics detailing total throughput estimates and the actual credit of the top 10% of peers (top 5000 by credit) we can calculate an estimate of the amount of new credit accumulated by the top peers as a proportion of all new credit created over time.

The statistical data was collected by scraping the web pages of the tracker. We have removed all invalid data due to server failures from our analysis. Because the amount of invalid data is small, it does not affect our analysis.

Table 4.1: Statistics from a popular private tracker over 7 days. (T=total throughput of the system, Δ=total credit increase per day, Δ_0=increase in credit of top 10% peers, δ=minimum fraction of total credit that goes to top 10% peers, and S/L=ratio of seeding to leeching sessions over the entire set of swarms.)

Day	T	Δ	Δ_0	δ	S/L
1	48	24	17	0.23	26
2	40	20	15	0.25	26
3	50	25	12	0.16	25
4	67	33.5	17	0.17	25
5	52	26	19	0.24	25
6	46	23	15	0.21	25
7	87	43.5	17	0.13	25
Ave.	56	28	16	0.19	25

In Table 4.1, T shows the total throughput of the system. Since we assume the system is closed, it holds that the total throughput is equal to both the total upload (U) and total download (D) in the system. Δ is the total credit increase in the system per day. The total credit increase comes solely from the *credit bonus* of 0.5 rewarded to uploading. Therefore, $\Delta = \frac{1}{2}U$.

Δ_0 is the increase in total credit of the top 10% of the population (in Terabytes). δ represents the minimum fraction of the total credit that goes to this top 10% (We explain how we arrive at this lower bound in the next section). S/L shows the ratio of seeding to leeching sessions over the entire set of swarms served by the tracker. We also calculated the turnover of the top peers which indicates how many of the top 10% of peers, by credit, change each day. We found this to be minimal, averaging 0.2% over each day (not shown in the table).

Note we can see that the ratio of seeding to leeching sessions (S/L) is high. This means that many peers are seeding much content - presumably to earn credit. Yet the top 10% of peers take a large proportion of the new credit created in the system (δ). We conjecture that this level of credit increase in the top peers is a result of high upload bandwidth and the high S/L is a result of credit starved peers seeding many swarms. Hence this evidence is consistent with the notion of a *credit squeeze* but not directly indicated by it since we do not know that adding more credit would improve throughput.

4.3 BitCrunch model description

In order to explore the minimal conditions under which certain credit dynamics occur, we have designed a model (BitCrunch) containing the essential properties of credit systems. We stripped away the complexities of real communities, so that the underlying forces of credit become clear and can be analyzed.

4.3.1 Peers

The community is represented by a set of peers. Each peer i has a predefined and fixed upload (up_i) and download ($down_i$) capacity (in units of data per unit of time).

Here we use a highly simplified user / client model. Peers are online at all times. At any given time a peer is seeding some number of swarms (S) and downloading from some number of other swarms (L). When a peer has finished downloading a file it moves it from it's download list to its seeding list. Peers seed files for some predefined and fixed amount of time and then remove them from their seeding list.

If the number of currently downloading files is less than L then the peer selects new swarms to download until L number of files are being downloaded. If adding a new seed to the seeding list causes the size to exceed S then the oldest seeding file is removed from the list. In this way each peer will always be downloading from L swarms and seeding a maximum of S swarms.

In our initial experiments we use a minimal form of this scheme by setting $L = S = 1$ and the maximum seeding time set to infinity. This means that each peer is always downloading in one swarm and seeding in one other swarm.

4.3.2 Swarm capacity

The tracker supports a set of swarms. Each swarm contains some number of seeder and leecher[2] peers (including possibly zero). Each seeder holds a copy of the entire file which the swarm is distributing. Each leecher contains some proportion of the file. Since every peer is assigned an upload and download rate we can calculate the total demand (sum of all leecher download rates) and the total supply (sum of all seeder upload rates). Hence for a given swarm S_i:

$$supply(S_i) = \sum_{j \in S_i} up_j$$

$$demand(S_i) = \sum_{k \in S_i} down_k$$

[2]We use the term leecher and downloader synonymously in this chapter.

where j is a seeder peer in S_i with up_j upload capacity and k is a leecher peer in S_i with $down_k$ download capacity.

In order to create transactions based on supply and demand we adopt a highly simplified swarm model. If supply matches demand then all leeching peers receive their entire download capacity and all seeders use their entire upload capacity. If demand exceeds supply (under supply) then all seeders use their entire upload capacity but each leecher k only receives:

$$download(k) = \frac{supply(S_i)}{demand(S_i)} * down_k$$

Hence if the demand was twice the supply then each leecher would only receive half of its download capacity. Conversely, if supply exceeds the demand then each leecher receives its entire download capacity but each seeder j uploads only:

$$upload(j) = \frac{demand(S_i)}{supply(S_i)} * up_j$$

Again this means that if supply was, say, twice the demand then each seeder would use only half its upload capacity.

As will be seen later our initial simulation experiments assume all peers have equal upload and download capacities meaning all peers in the same swarm obtain identical service.

4.3.3 Ratio enforcement

All peers begin with an equal amount of credit which for all peers in the system sums to C. When peers download, their individual credit score is reduced accordingly. Conversely, uploaders have their credit score increased. In this way C stays constant over time but the distribution, over peers, changes depending on download and upload behaviour.

If a peer runs out of credit it can no longer download and is excluded from the list of active downloaders in any swarm it is trying to download from. When a peer enters this state it is considered “broke” and cannot download again until it earns credit from its seeding activities in other swarms.

4.4 Simulation experiments

To examine the relationship between initial credit, efficiency, and credit dynamics in our model we produced two sets of simulation runs. In the first set of “baseline” runs we initialized each peer with equal upload and download capacities and credit. No new credit was created over time. In the second set of ”unequal capacity” runs we added capacity to

the system by setting a minority of peers to have higher upload capacity than the majority and, in addition, we experimented with creating new credit in the system by awarding a "seeding bonus".

The simulation runs proceed in discrete time units (or cycles). One time unit involves each swarm transacting uploads and downloads based on the supply / demand model previously described.

4.4.1 Baseline runs

We started each run with all peers equally sharing the initial credit C and having equal upload and download capacities. All swarms were assigned equal popularity and each file had the same size. We defined efficiency as how much data was exchanged in units (throughput) over a fixed time period.

We set all file sizes to 10 units. We set the upload and download capacities of each peer to 1 unit (per time unit). We performed runs for three different initial credit values (1, 10, and 100 per peer). We set the number of peers to 500 and the number of swarms to 100. This ensures sufficient peers to create meaningful levels of supply and demand in each swarm.

These values give a highly balanced baseline in which all things are equal for each peer. In the real world of private trackers such balance would not occur since peers have different upload / download capacities, availability, and user behavior. In addition swarms follow non-equal popularity distributions [4]. We performed these simulations to determine if credit squeeze phenomena can be observed in our simple model where key parameters are held equal.

Table 4.2 shows results from simulation runs with different initial credit amounts (per peer) given by C. Each value represents an average of 10 simulation runs to 2000 cycles (variance was negligible). T is the total cumulative throughput at the end of the runs. This is shown as a proportion of the maximum throughput that could be achieved if no peer is allowed to become "broke" - i.e. if all peers have infinite credit. As can be seen T increases as C increases. β gives the proportion of peers with zero credit (broke peers) at the end of each cycle averaged over the last 10,000 time units. G gives the *Gini inequality measure*[3] for peer credit at the end of each cycle - again averaged over the last half of the runs. φ gives the turnover of the top 10% of peers (by credit) as a proportion. This is calculated after every 100 time units and is averaged over the last half of the runs. Hence turnover (φ) gives a measure of the "credit mobility" of peers as a proportion of change in the top 10% of peers over time.

Given our definition of a credit squeeze it is evident that when initial credit (C) is low

[3]The Gini coefficient takes a value from $[0, 1]$ and characterizes inequality, with 1 being the most unequal (one peer holds all credit) and 0 being complete equality.

Table 4.2: Results from baseline simulation runs. (C=initial credit, T=system throughput, β=proportion of broke peers, G=credit inequality, φ=Credit turnover.)

C	T	β	G	φ
1	0.58	0.36	0.87	0.84
10	0.81	0.20	0.77	0.43
100	0.97	0.06	0.59	0.10

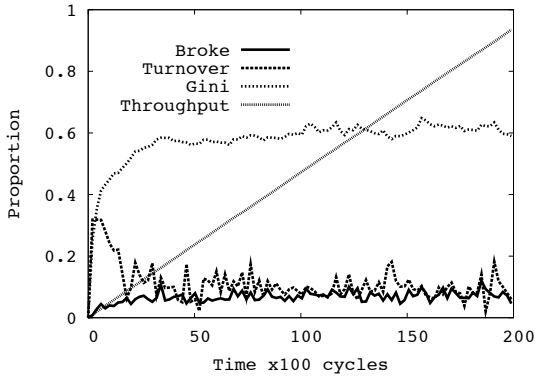


Figure 4.1: Typical time series of a single simulation run of the model where initial credit is set to 100 units per peer with peers having equal capacity.

then throughput (T) is also low. However, when $C = 100$ units of credit per peer then T is 94% of the maximum T achievable. Notice also that peer credit inequality (G) decreases as C increases but that even when $C = 100$ there is still high inequality with a Gini value of 0.59. Turnover (φ) also decreases as credit increases. However, even when $C = 100$, turnover is still high implying that the top 10% of peers by credit could change completely within 1000 time units.

Figure 4.1 shows a typical time series from a single simulation run for C=100. Note that, after an initial period, values settle within stable bounds. Because the proportion of broke peers (β) is low, due to high initial credit, the total cumulative throughput (T) almost reaches the maximum throughput found when credit was infinite.

4.4.2 Unequal upload capacities

In order to examine the credit dynamics when some peers have differing upload and download capacities we performed a further set of simulation runs in which 10% of peers were

Table 4.3: Results from unequal upload capacities simulation runs. (C=initial credit, T=system throughput, β=proportion of broke peers, G=credit inequality, φ=Credit turnover.)

C	T	β	G	φ
1	0.56	0.39	0.90	0.82
10	0.71	0.32	0.93	0.44
100	0.77	0.29	0.94	0.60
100++	0.97	0.01	0.71	0.00

initialized with upload and download capacities of 10 units. The other 90% were given a download capacity of 10 units but an upload capacity of only 1 unit. Here we capture the notion that only a small number of peers have high upload capacity whereas all peers have high download capacity. We kept all other parameters the same as in the baseline runs.

Table 4.3 shows the results obtained. As could be expected we see a high level of credit squeeze for the fixed credit runs (C=1, 10 and 100). Notice however that for the C = 100++ runs we have a minimal credit squeeze effect. In the C = 100++ runs we initialized each peer with C = 100 but implemented a "seeding bonus" scheme in which seeders receive an additional 50% credit bonus on all upload. This means a seeder uploading 1 unit receives 1.5 units of credit.

Interestingly we found that the throughput value, T, was actually lower in the fixed credit scenarios[4] even though substantial capacity had been added to the system (in upload and download).

Figure 4.2 shows a typical time series from a single simulation run for C=100. Note that after an initial period values settle within stable bounds. Notice that the proportion of broke peers (β) is much higher than the previous baseline time series and hence the cumulative throughput (T) of the system is reduced.

4.5 Discussion

From the empirical results in Section 4.2 we conjectured evidence of a credit squeeze. Our simulations results show that *even in a trivial model where all peers have the same capacities and user behaviour, all swarms have equal popularity and all peers start with equal credits, the performance of the system may be inhibited by credit shortages*. This is because high levels of credit skew emerge due to the fact that a peer can only upload a

[4]By this we mean that even if T is expressed as an absolute value, rather than as a proportion of maximum capacity, it is lower than in the baseline runs

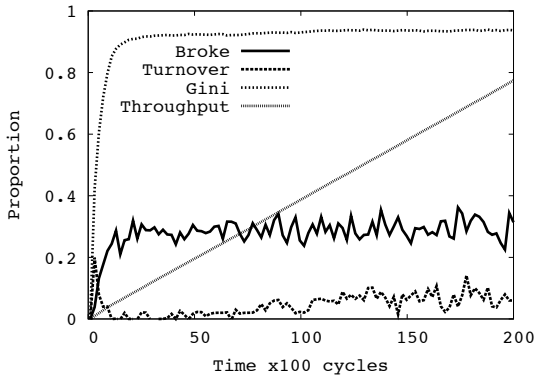


Figure 4.2: Typical time series of a single simulation run of the model where initial credit is set to 100 units per peer with peers having unequal capacities.

file it has already downloaded.

We also observe that in such scenarios *adding extra capacity to the system, in the form of upload and download capacity, can actually reduce the performance*. This is highly counter intuitive and something that should be avoided because it implies lack of scalability.

Finally, we found that *by injecting new credit into the system in the form of a "seeding bonus" a credit squeeze can be ameliorated* when peer capacities are unbalanced.

The results from our empirical evidence and model must be qualified because we have excluded credit earned during TFT behavior - where downloading peers exchange data with other downloaders. Based on our private tracker measurements we have concluded that the majority of exchange is between seeders and leechers but we cannot be sure of this [5].

Additionally our assumption that swarms resolve supply and demand in an equitable way is not the case in BitTorrent due to limited upload "slots" and the randomised nature of peer selection over time. Our abstraction is that over time and swarms this might be comparable to equal shares. But again we cannot be sure of this.

In our model we observed that our swarms became quickly skewed, containing many leechers and a small number of seeders and vice versa. Why does this happen when selection of swarms is uniformly random? This is because any initially small imbalance becomes exaggerated through positive feedback. A swarm with high leecher / seeder ratio gets slow and hence tends to "recruit" more leechers. A swarm with a low ratio will tend to turn new leechers into seeders very quickly, creating more seeders. But this would

[5] As of writing this chapter, recent measurement studies have emerged that confirm this hypothesis [79]

not happen in BitTorrent because of TFT effects (which our model excludes). Although, coincidentally, the flash-crowd swarm life-cycles observed in real systems could produce similar skews [4].

Since we have not put freeriders into our model, we have not explored the tradeoff between amount of credit in the system and de-incentivizing freeriding. A freerider would take as much credit from the system as they could without reciprocating. This could be an interesting area for future research.

4.6 Related Work

A large amount of work has been done on studying the BitTorrent protocol and communities. Many research studies have been conducted to determine the robustness, scalability and performance of BitTorrent-like systems [9, 56, 60, 66, 72, 93, 121, 127].

It should be noted that these works focus on the single swarm while we are concerned with the multiple swarm scenario. Other works which have focused on multiple swarms address the so-called 'seeder promotion problem'. The basic claim across these papers is that while the TFT policy of the BitTorrent protocol could be considered practical for single swarms, it does not offer incentives for peers to seed content after finishing their own downloads [36, 102, 117]. All these works include a kind of monetary mechanism to provide incentives for cooperation among peers.

A recent work by Andrade et al. [4] examines the supply and demand for resources in public and private BitTorrent file sharing communities. Their main contributions include showing that only a minority contributes the majority of the resources and that the upload contribution of peers is not correlated to the time that they invest in seeding content. These findings lend credence to our hypothesis that a minority of the users hogs the majority of the credits while a large majority are unable to earn credit despite seeding content for long durations.

Credit crunches and crashes have been studied in Scrip Systems by Kash et al. [62]. They show that both an overabundance of money supply and its shortage lead to inefficiency. An overabundance in the money supply leads to a monetary crash where no one is willing to work and freeriding is encouraged. On the other hand, a shortage in the money supply leads to peers going broke and not being able to afford services in the system. This work is different from ours in that we place our analysis firmly in the very practical domain of private BitTorrent communities and bring forth real life measurements to support our hypothesis.

4.7 Conclusion

We have introduced the notion of a "credit squeeze" where lack of credit impacts the efficiency of private tracker systems. We presented some initial statistics from a popular private tracker that are consistent with this idea but are not conclusive.

We defined a simplified model of a private tracker and found that insufficient initial credit can lead to a credit squeeze even when all peers have equal capacities, user behaviour, and all swarms in the system have equal popularity. Through simulation we showed that in such systems highly skewed credit distributions emerge between peers but that over time peers have high mobility of credit rank. This means the credit rich do not stay rich and poor do not stay poor but at any given time there are rich and poor.

We also performed simulations where a minority of peers had much higher upload bandwidth capacity than other peers. Here we found that the minority became highest in the credit rank, as to be expected, but the imbalance led to high system level inefficacy due to a credit squeeze in the lower capacity peers. Hence adding capacity to the system actually reduced efficiency because the rich minority, over time, accumulate the majority of credit in the system. Interestingly, we found that by applying a policy modeled on that used by the private tracker, from which we collected statistics, solved the credit squeeze at the expense of reduced rank mobility (it went to zero). This policy involved giving seeders a "bonus credit" of 50% of their upload. Essentially this equates to imposing ratio of at least 2/3 on all peers (i.e. they only need to upload two thirds of what they download to stay in credit).

The work presented in this chapter is at an early stage. We have a hypothesis that a lack of credit circulation, due to a minority of peers obtaining the majority of the credits, can severely degrade the efficiency of private trackers. However, while we do have evidence from a single private tracker that the gap between the rich peers and slow peers is widening, we did not have access to the relevant data that could prove that this phenomenon also hampers the efficiency of such systems.

However, using a simplified model of a private tracker, we were able to show through simulation that the disparity of wealth (credits) among the peers leads to a credit squeeze. In the model, we exclude many phenomena that could lead to credit skews such as: injecting of new content; differing swarm popularity and swarm "life-cycles"; peer availability and differing user behavior (including freeriding). We believe all of these factors are highly important in shaping how private trackers with credit based incentive systems, actually perform.

Chapter 5

Sustainable credit dynamics in a P2P community

As explained in the previous chapter, the TFT approach in BitTorrent does not provide incentives for a crucial activity called "seeding". A seeding peer stores the entire file and hence acts in a purely altruistic way by giving away pieces. To create a new BitTorrent swarm at least one seeder is required. When a peer has downloaded the entire file it automatically becomes a seeder unless the user of the peer decides to leave the swarm. A swarm containing many seeders provides high download rates for peers downloading from the swarm (termed "leechers"). Yet since seeding is not incentivised many swarms suffer from so-called "Hit and Run" (H&R) user behaviour where peers leave the swarm after downloading the file.

In order to address these and other issues, private BT file-sharing communities have recently emerged. In many such communities the upload and download behaviour is recorded centrally, over time, across a population of swarms that are restricted to the community. Many of these communities apply policies to incentivise good overall upload / download behaviour - such as ratio enforcement. For example, a simple policy would be to expect all peers to have some minimum ratio of upload to download at all times. Other policy variants include requiring some minimum level of absolute credit (upload - download) over time, or detecting and punishing H&R behaviour.

In Chapter 4 we showed that even with altruistic users, in which all peers seed as much as possible, private community credit systems could, counter-intuitively, lead to poor performance due to a "credit crunch" or squeeze in which a few peers accumulate much of the credit in the system, depriving others and hence decreasing overall system throughput. This means that adding altruistic capacity to the system, in the form of high capacity peers who are willing to upload without reciprocation, can actually reduce overall

performance (or throughput) - meaning the total amount of data exchanged in the system[1]. It should be noted that private BitTorrent communities regularly employ policies such as 'seeding bonus', 'free leech', rewarding seeding time instead of bandwidth, among other schemes. The fact that various private BitTorrent communities have to employ such policies is indicative of the fact that they are grappling with performances loss due to credit mobility issues.

In this chapter we explore credit dynamics with two user model variants: *selfish*, where peers only contribute what is necessary to allow them to continue to download content, and *hoarders*, where peers desire to accumulate more credit than is necessary for their immediate downloading needs.

We find for populations of selfish peers, when too much credit is distributed too evenly, this leads to a crash in which peers are not incentivised to contribute and hence the system seizes to zero throughput containing only leechers. *We define a crash as a situation in which due to credit abundance the system completely seizes up, providing no upload or download to any peers*. Conversely, too little credit distributed over the peers leads to a crunch in which peers do not have enough credit to download leading to a seized system containing only seeders. *We define a crunch as a situation in which due to credit shortages the system completely seizes providing no upload or download to any peers*. We also observe that a population containing any number of hoarders will lead to a crunch eventually as credit is monopolized by them.

Specifically, we make the following contributions: (i) we demonstrate using simulations that credit crunches and crashes can occur, and identify the conditions that lead to these extreme outcomes; (ii) we present a theoretical analysis and produce a set of propositions that define when a system will crash, crunch, or be sustainable over a defined time horizon; (iii) we propose and evaluate a novel adaptive credit policy informed by our results and analysis. We demonstrate that this adaptive policy can avoid both crashes and crunches under extreme and changing conditions.

5.1 Model Description

In order to explore the conditions under which credit crashes and crunches occur we designed an agent-based simulation model containing the essential properties of private tracker credit systems.

[1]This can be compared to Braess Paradox in which adding capacity to transport networks may reduce total flow under the assumption of rational actors – see: http://en.wikipedia.org/wiki/Braess's_paradox

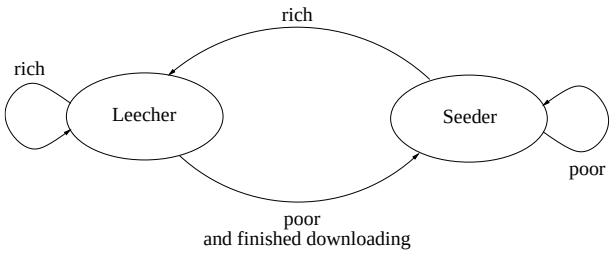


Figure 5.1: Seeding and Leeching Sessions.

We abstracted away the particularities of real communities, so that the underlying credit dynamics become clear and can be analyzed.

Our simulation model is based on *cycles*. Each cycle represents a unit of time in which each peer is activated and may perform some activity - such as uploading or downloading data from other peers and initiating new seeding or leeching sessions. Peers accumulate and spend credit by participating in swarms which are supported by a Tracker. The Tracker keeps a record of all current swarm members and records the upload and download amounts against each peer. We describe the tracker, swarm, and peer entities in more detail in the following subsections below.

5.1.1 Tracker

The tracker supports a set of swarms that are available to the community and stores the upload and download amounts reported by each peer over time. The tracker implements a ratio enforcement policy in which peers with upload / download ratio less than one are stopped from downloading content until they increase their ratio, through seeding content in a swarm. In order to allow peers to begin downloading, they are awarded initial credit equal to one file size. It should be noted that the tracker in our model, like in private BitTorrent communities, is a centralized component.

5.1.2 Peers

The community is represented by a set of peers. Each peer has the same fixed upload capacity, representing units of data per time unit. Download capacity is assumed to be infinite - the assumption being that upload is the bottleneck in most file sharing communities. We make a further simplifying assumption that peers have a maximum of one seeding or one leeching session active at any time.

We implement two user types: selfish peers and hoarders. User types are characterized by their leeching and seeding behavior. Selfish peers only seed (to earn credit) if their current credit balance is less than one file size (C) otherwise they only leech. This captures the notion that a selfish peer only wants enough credit to download the next file. Hoarder peers behave in a similar way but have a higher threshold before they stop seeding.

More formally, we define two functions $u(t)$ and $d(t)$, which are the total number of units a peer has uploaded and the total number of units a peer has downloaded at time t, respectively. The credit of a peer at time t is $K(t) := u(t) - d(t)$, while the ratio is $R(t) = u(d)/d(t)$. Peers can always be considered in one of two states: "rich" or "poor". Peers are considered to be "poor" except under the following conditions: a selfish peer is "rich" when $K(t) \geq C$; and a hoarder peer is "rich" when $R(t) \geq 2$.

When a leeching peer finishes downloading a file it decides whether to seed that file or to select another file to leech (i.e. to download) from the set of swarms available. If a peer is rich it selects a new swarm with uniform probability and begins to leech without seeding the previously downloaded file. If a peer is poor then it seeds the downloaded file until it becomes rich and only then stops seeding and starts leeching in another swarm. Figure 5.1 shows a state transition diagram indicating how peers move between leeching and seeding states.

Our user model represents an abstracted form of behaviour compared to what is observed in real file sharing communities. Figure 5.2 shows the cumulative distribution of peer ratio over a four month period from a real community. It can be observed that the majority of peers are within one order of magnitude of ratio $R(t) = 1$ where t is the end of the period. Notice here that we also observe a small proportion of peers with very high ratios where $R(t) > 10$, indicating some extreme hoarding behaviour.

5.1.3 Swarms

The community comprises a set of swarms S. Each swarm may contain any number of seeders and leechers[2]. At any time each peer will be either a seeder or a leecher in a single swarm. In one time unit each swarm distributes upload units (pieces) between seeders and leechers and between leechers and leechers. The latter distribution of upload captures the effect of the tit-for-tat (TFT) mechanism in BitTorrent. We do not assume any freeriding or differential upload capacity at the piece level. All peers utilize their full upload to contribute to others and all have equal upload capacities.

[2]We ensured that all swarms have at least one seeder - if this is possible - by randomly redistributing a seeder to any swarm that becomes seederless.

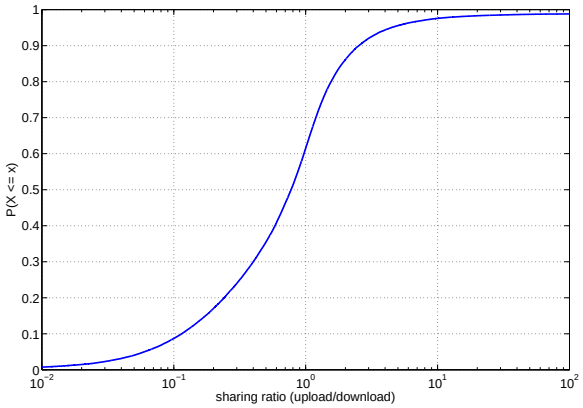


Figure 5.2: CDF of Peer Sharing Ratios.

In more detail, we model the distribution of upload in a swarm in the following way. In our model a file is divided into units (or pieces in BitTorrent terminology). Similar to many BitTorrent implementations, each peer has four upload slots from which it can upload data to four different leeching peers per time cycle. At every time cycle, each peer in a swarm is fired in random order. When a peer is activated it chooses four leeching peers at random to upload data to. In line with the *rarest piece first* [66] selection algorithm in BitTorrent, peers upload the rarest pieces to other peers first. We define rarest pieces to be the least replicated pieces among all the peers in a swarm. Hence all pieces are ordered by rarity and each of the four receiving peers is given the rarest piece that it does not currently posses. In the case that a receiving peer already has all the pieces available in the sending peer then another peer is selected randomly to receive a piece. In this way each peer will send one piece to four other randomly selected peers in the swarm if this is possible.

It is important to note that our aim is to produce a simple model that captures the main characteristics of BitTorrent piece sharing under constant and ideal conditions in order to investigate the importance of credit dynamics. In the real world the actual interactions between peers would be highly influenced by differential bandwidths, different clients, and the local nature of the piece rareness determination. Our aim here is not to produce simulations that align with measurements from target systems but to abstract the important mechanisms with respect to credit dynamics.

As stated previously, the tracker records all upload and download against each peer and hence keeps a running total of credit and ratio for each peer.

Table 5.1: Results for Selfish Peers with Constant Credit.

prop.of **rich** at start	avg.**throughput** (std.dev)	avg.prop.of **seeders** (std.dev)	final **state**
0.1	0.000 (0.000)	1.000 (0.000)	crunch
0.3	0.218 (0.001)	0.953 (0.001)	sustain
0.5	0.777 (0.002)	0.769 (0.002)	sustain
0.7	0.968 (0.004)	0.506 (0.004)	sustain
0.8	0.587 (0.478)	0.249 (0.478)	sustain/crash
0.9	0.001 (0.000)	0.000 (0.000)	crash

5.2 Simulation Results - Constant Credit

We performed a number of simulation experiments to explore the conditions under which a sustainable file sharing community is viable under the assumption of a fixed credit amount in the population. We used the following parameters: number of peers $N = 1000$, number of swarms $s = 100$, file size $C = 10$ units, peer upload capacity $U = 4$ units. The small file size means the simulation runs produce results at a large scale of granularity. We also performed runs with $C = 100$ and found no significant difference in results. In general our results are independent of the size of C. As stated previously, we assume no limit on download capacity. For each experiment we performed 10 independent runs with different pseudo-random number seeds. Each run was executed to 2000 cycles.

5.2.1 Populations of selfish peers

In order to explore the results of different initial credit levels on the performance of populations containing all selfish peers we ran several simulation experiments varying a single parameter - the initial proportion of peers who are given enough credit to be "rich" (i.e. given initial credit of C).

Results can be seen in Table 5.1. All values are averages of 10 runs. The columns have the following meanings: *rich* shows the proportion of peers that are initially awarded C credit; *throughput* gives the throughput of the system in cumulative units of data exchanged over the entire run (normalized); *seeders* shows the proportion of peers in the population at the end of the run that are seeding; *state* indicates the state of the system: *crunch* means the system seized due to lack of credit and *crash* indicates seizure due to too much credit. *Sustain* means the system finds a stable sustainable throughput avoid-

ing both crashes and crunches. We ran extended runs up to 20,000 cycles and found the sustainable outcomes were maintained.

When the number of rich peers is initialized to 30%, 50% and 70%, respectively, we see sustainable outcomes with increasing throughput and reduced number of seeders. This is intuitive since as the amount of credit increases in the system less peers are poor and hence more exchange of data can occur.

Notice that in the crunch state, where only 10% of peers are initialized as rich, the system is composed of all seeders by the end of the run and hence no exchange of data can occur. Conversely, in the crash state, where 90% of peers are initialized as rich, all peers are leechers by the end of the run, again, meaning no exchange of data is possible. Inspection of individual runs evidences that crunches and crashes happen quickly - within the first ten cycles or so. This is reflected in the low (almost zero) cumulative throughput values under crash and crunch states.

It is interesting to consider the results when the initial number of rich peers is set to 80%. As can be seen this produces both sustain and crash outcomes reflected in the high variance of throughput. We are not sure why exactly this happens. But what is obvious is that here we are very close to the threshold leading to a crash and we find path dependency based on initial random conditions leading to either a high sustainable throughput, in a third of runs, or a sudden crash otherwise. Figure 5.3 shows the two alternative trajectories. Such properties are common in complex systems where small differences in initial chance conditions can lead to radically different outcomes.

5.2.2 Populations containing hoarder peers

When we introduced *any* number of hoarder peers into our system it eventually led to a crunch - this is true even for a single hoarder in the population. The speed of the crunch depended on the number of hoarders. This is intuitive since hoarders seed (to earn credit) until they have a ratio $R(t) \geq 2$. This means that as the simulation progresses the hoarders eventually hold all the credit in the system and a crunch is inevitable.

5.2.3 Discussion

The results of our initial experiments indicate that a community will seize if all the swarms seize. A seized swarm is one that will not allow any peer within it to leave. A peer can leave either by downloading the entire file and then moving to another swarm (if the peer

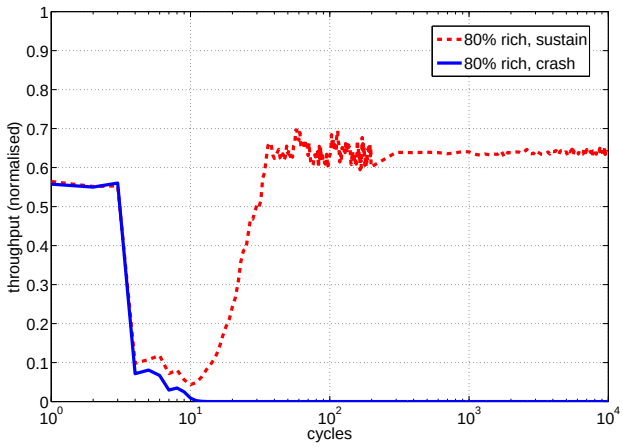


Figure 5.3: Two Alternative Trajectories (Initial Number of Rich Peers is Set to 80%).

finishes the download and remains rich) or by seeding to earn enough credit to become rich so that it can move to another swarm for leeching. In the crunch condition there will not be enough leechers to satisfy seeders and in the crash condition there will not be enough seeders to satisfy leechers. In these cases the swarm becomes a kind of trap or sink that, unless some new peers with certain characteristics enter the swarm in the future, stops the peers from further exchange in other swarms.

Given the state of a swarm, including all peer credit levels at some time step, it is possible to define if the swarm *will* seize in the future due to a crash or crunch condition - unless something changes. In the next section, we present a theoretical analysis which specifies these conditions. Following this we validate the model by checking that these conditions apply to crashes and crunches. We then apply the conditions to test a novel mechanism for automatically adapting credit policies when the system is identified as heading for crash or crunch conditions in order to avoid them.

5.3 Theoretical Results

We wish to determine for a given set of swarms if the system will crunch, crash, or be sustainable. We assume a peer can either leech or seed, exclusively, and only exist in one swarm at one time. Firstly we define some terms and definitions and then present three propositions which give conditions for each of the three outcomes.

The following notation is used: S_ℓ is the swarm ℓ, where $\ell = 1, \ldots, s$, i.e. the number

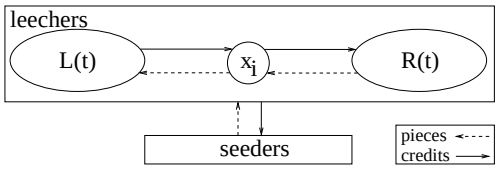


Figure 5.4: Relationship Between L, R and x.

of swarms is s; the number of leechers and seeders in swarm ℓ at time t is $x^\ell(t)$ and $y^\ell(t)$, respectively; x_i^ℓ is the leecher i and y_j^ℓ is the seeder j in swarm ℓ; $c_{x_i^\ell}(t)$ is the credit of leecher i and $c_{y_j^\ell}(t)$ is the credit of seeder j in swarm ℓ at time t; $p_{x_i^\ell}(t)$ is the proportion of the file that leecher x_i^ℓ has at time t; C is the amount of credit required to download a file (i.e. the file size); and u is the upload bandwidth.

Moreover, define $L_i^\ell(t)$ as the set of leechers which have less pieces of the given file, represented by swarm ℓ, than leecher x_i^ℓ, and $R_i^\ell(t)$ is the set of leechers which have more pieces of the given file, represented by swarm ℓ, than peer x_i^ℓ at time t. Formally, $L_i^\ell(t) := \{j \ : \ p_{x_j^\ell}(t) < p_{x_i^\ell}(t)\}$ and $R_i^\ell(t) := \{j \ : \ p_{x_j^\ell}(t) > p_{x_i^\ell}(t)\}$. Figure 5.4 shows a diagram of these sets. It can be seen from the figure that we make the simplifying assumption that peers can only download pieces from those peers that have more pieces. Therefore, a peer can only give credit to those peers who have more pieces. It should be noted that in reality, this is not the case. If two peers have downloaded complementary parts of a file, they can exchange pieces between themselves no matter who has more pieces. However, the results in Section 5.4 demonstrate that this assumption does not impede upon the predictive powers of our subsequent analysis.

We also need to define the amount of credit that any given leecher can earn from other leechers:

$$q_{x_i^\ell}(t) := \sum_{j \in L_i^\ell(t)} (p_{x_i^\ell}(t) - p_{x_j^\ell}(t)).$$

Given the above, we can now define a set $X_\ell(t)$ containing those leechers with enough existing credit, plus credit earning potential, at time t, to download the entire file associated with swarm ℓ and still be in a rich state. As previously defined, a peer is considered rich if it has credit of at least C (i.e. one file size). Leecher x_i^ℓ will be able to download a file and still remain rich if $c_{x_i^\ell}(t) - (1 - p_{x_i^\ell}(t)) \times C \geq C$ holds. During the download x_i^ℓ is not only paying but also earning some credits from other leechers in the swarm ℓ, from those who have less pieces than x_i^ℓ. From these peers x_i^ℓ can earn $q_{x_i^\ell(t)} \cdot C$ amount. On the other hand, x_i^ℓ cannot earn credits from those who are seeding and who have more pieces. The number of those peers are $y^\ell(t) + |R_i^\ell(t)|$. Thus, from the other leechers, x_i can earn

$q_{x_i^\ell}(t)/(y^\ell(t)+|R_i^\ell(t)|)\cdot C$ amount of credit. This leads to the following definition of $X_\ell(t)$:

$$X_\ell(t) := \{x_i^\ell : c_{x_i^\ell}(t) + p_i^\ell(t)C + \frac{q_{x_j^\ell}(t)}{y^\ell(t)+|R_i^\ell(t)|}C \geq 2C\}.$$

Note that when a leecher finishes its download and becomes poor then it stays in the same swarm to seed. In this case X_ℓ remains the same, because $y+R_i$ does not change. According to our assumption no new seeders can join to this swarm from other swarms. X_ℓ can be changed when new leechers join the swarm as they can give the other leechers the chance to become rich after their download (X_ℓ could increase then), or when seeders are leaving the swarm (X_ℓ could decrease).

Similarly, we define the set $Y_\ell(t)$ for seeders in the swarm ℓ containing those seeders which have the credit earning potential, at time t to become rich. A seeder earns credit from all the leechers. One leecher x_i^ℓ gives $(1-p_{x_i^\ell}(t))C$ amount of credit to all the seeders and to those other leechers who have more pieces than x_i^ℓ. Thus we define $Y_\ell(t)$ as:

$$Y_\ell(t) := \{y_j^\ell : c_{y_j^\ell}(t) + \sum_{k=1}^{x^\ell(t)} \frac{1-p_k^\ell(t)}{y^\ell(t)+|R_k^\ell(t)|}C \geq C\}.$$

Note that, similarly to X_ℓ, the set Y_ℓ can be changed in time only when new leechers are joining (Y_ℓ could increase) or when a seeder is leaving the swarm (Y_ℓ could decrease then).

We now introduce a temporal horizon by defining a function for download time for leechers and required seeding time for seeders.

The expected download time $T_{x_i^\ell}$ for the leecher x_i in the swarm ℓ depends on the remaining amount to be downloaded (which is $(1-p_{x_i^\ell}(t))C$) and on the number of seeders and leechers in the swarm. The seeders can give pieces to all the leechers with the rate of $y^\ell(t)/x^\ell(t)\cdot u$. Moreover, every leecher x_k, which has more pieces than x_i, i.e. the peers in R_i can give pieces to those who have less pieces than x_k, which is the set L_k. The rate of this is $\sum_{k\in R_i} 1/|L_k|\cdot u$. Combining these observations we get for $T_{x_i^\ell}(t)$:

$$T_{x_i^\ell}(t) := t + \frac{(1-p_{x_i^\ell(t)})\cdot C}{(\frac{y^\ell(t)}{x^\ell(t)} + \sum_{k\in R_i^\ell(t)} \frac{1}{|L_k^\ell(t)|})\cdot u}.$$

Note that $T_{x_i^\ell}$ is only the expected time given current swarm composition. The actual time could be longer if new leechers joined or seeders left the swarm. We also need the expected time $T_{y_j^\ell}$ for a seeder y_j in swarm ℓ to become rich. A seeder, at time t, needs to earn $C - c_{y_j}^\ell(t)$ credit. The rate at which this can be obtained depends on the total upload

capacity of all leechers less the leecher to leecher (TFT) interactions. This leads to the formula:

$$T_{y_j^\ell}(t) = t + \frac{C - c_{y_j^\ell}(t)}{\sum_{i=1}^{x^\ell(t)} \frac{1}{y^\ell(t)+|R_i^\ell(t)|} u}.$$

Note that $T_{y_j^\ell}$ is also only the expected time given the situation in the swarm at a time instance.

Theorem 5.1 - Crunch. If $X_\ell(t')$ and $Y_\ell(t')$ are both empty for all $\ell = 1, \ldots, s$, then the system will crunch (i.e. the throughput becomes zero) at time

$$t' + \max_{\ell=1,\ldots,s} \max_{i=1,\ldots,x^\ell(t')} T_{x_i^\ell(t')}.$$

Proof. For the sake of simplicity we omit t' and ℓ from the formulas. If X and Y are both empty for all the swarms, then it means that there are no leechers and no seeders that become rich. Thus, there will be no exchange of credit in the whole system. This happens after the very last leecher finishes its download, which is the maximum of all the maximum of download times per swarms. □

Theorem 5.2 - Crash. If $|Y_\ell(t')| = y^\ell(t')$ and

$$\min_{k \in P_\ell(t')} T_{x_k^\ell}(t') > \max_{j=1,\ldots,y^\ell(t')} T_{y_j^\ell}(t')$$

for all the swarms, where $P_\ell(t) := \{i \ : \ x_i^\ell \notin X_\ell(t)\}$, then the system will crash (i.e. the throughput becomes zero) at time

$$\max_{\ell=1,\ldots,s} \max_{j=1,\ldots,y^\ell(t')} T_{y_j^\ell}(t').$$

Proof. The system crashes if there are no seeders. The condition $|Y| = y$ indicates that all the seeders in the whole system will be rich, thus they will become leechers. The set P contains those leechers which will not be rich after finishing their download. Note that this set can be empty for some swarms. Peers from P would stay to seed in their swarm if they finished their download. However, they cannot finish downloading the file if there are no seeders left in the swarm. □

Theorem 5.3 - Sustainability. If the set

$$\mathscr{U}(t') := \{\ell \ : \ |X_\ell(t')| > 0 \text{ and } |Y_\ell(t')| < y^\ell(t')\}$$

is not empty, then the system is sustainable until

$$t'' = t' + \min\{M_{\mathscr{U}}, M_{\mathscr{V}}\},$$

where

$$\begin{aligned} M_{\mathscr{U}} &:= \min_{\ell \in U}\{\min_{x \in X_\ell(t')} T_x, \min_{y \in Y_\ell(t')} T_y\}, \\ \mathscr{V} &:= \{\ell \,:\, |X_\ell(t')| > 0 \text{ or } |Y_\ell(t')| < y^\ell(t')\} \text{ and} \\ M_{\mathscr{V}} &:= \min_{\ell \in V}\{\min_{x \in X_\ell(t')} T_x, \min_{y \in Y_\ell(t')} T_y\}. \end{aligned}$$

Proof. The set $\mathscr{U}$ contains all swarms in which there are some leechers that will be rich after their download and seeders that will not be leaving (i.e. not be rich), based on the current situations at time t'. Those seeders who are characterized, as of time t', as not leaving the swarm, could leave if new leechers join the swarm. This would happen when a leecher or seeder from another swarm leaves and joins this swarm. Thus, giving the time horizon for sustainability, we need to know the earliest time when the current situation could change. For this, we defined $M_{\mathscr{U}}$, which is the minimum time of the first leecher leaving any swarm in $\mathscr{U}$ and the first seeder becoming rich from any swarm in $\mathscr{U}$. Moreover, $M_{\mathscr{V}}$ gives the same for the swarms in $\mathscr{V}$. Finally, we need to take the minimum of $M_{\mathscr{U}}$ and $M_{\mathscr{V}}$ in order to get the prediction period for the system's sustainability. □

5.4 Simulation Results - Adaptive Credit

Based on our experimental and theoretical results, we have designed a novel *proactive credit intervention mechanism* to avoid crashes and crunches. At each cycle we examine the system and compare it against the crash and crunch propositions derived from the theoretical analysis. Hence we can obtain an early warning for potential crunch or crash outcomes.

When the system is determined to be entering a crunch, the system applies a new credit policy we term "freeleech" policy. This means leeching peers in the swarms do not pay any credit (no download is recorded) but seeders and other uploaders are still credited with upload. Hence leechers can download for free and new credit is injected into the system.

When the system is determined to be entering a crash, it applies a "freeseed" policy. This means seeding peers (and uploading leechers) in the swarm do not receive any

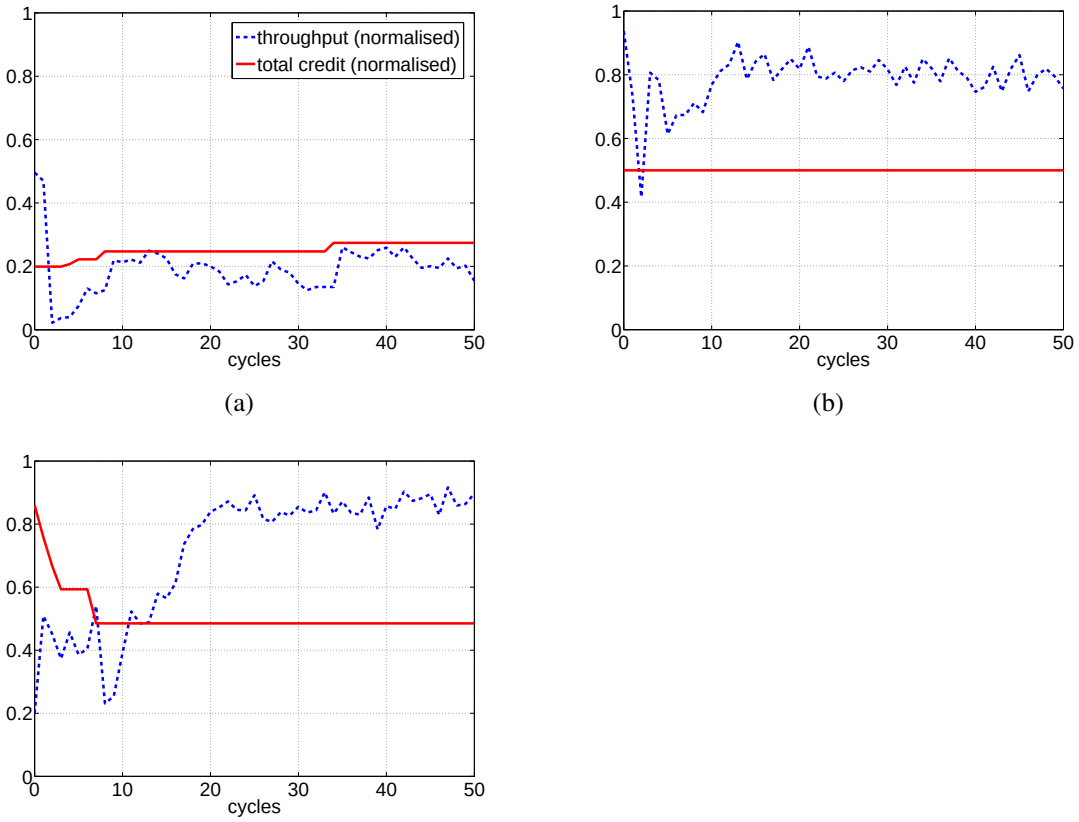


Figure 5.5: Adaptive Credit Mechanism.

credit (no upload is recorded) but leechers still pay credit to download. Hence seeders (and uploading leechers) upload for free and leechers pay credit that is removed from the system.

When the system is determined to be in a sustainable state then the regular credit policy is applied, that is, all upload and download is recorded as normal.

If one views uploaders as *producers* and downloaders as *consumers* then freeleech is rather like a 100% rebate for the consumer for any purchase and freeseed is like a 100% tax on the producer.

5.4.1 Populations of selfish peers

Table 5.2 shows the results of performing simulation runs with the same parameters as were used for the runs given in Table 5.1, but with the adaptive credit mechanism turned

Table 5.2: Results for Selfish Peers with Adaptive Credit.

prop.of **rich** at start	avg.**throughput** (std.dev)	avg.prop.of **seeders** (std.dev)	final **state**
0.1	0.192 (0.022)	0.956 (0.022)	sustain
0.3	0.222 (0.010)	0.954 (0.010)	sustain
0.5	0.777 (0.002)	0.776 (0.002)	sustain
0.7	0.967 (0.002)	0.497 (0.002)	sustain
0.8	0.974 (0.016)	0.530 (0.016)	sustain
0.9	0.880 (0.069)	0.727 (0.069)	sustain

on. Notice that all runs produce a sustainable outcome including those runs (10% rich and 90% rich) which previously led to crunches and crashes. This indicates that the theorems have given an early enough warning for the adaptive credit policy to avoid the crash and crunch conditions being reached. If the system ever enters a crash or crunch condition then it would seize completely and the adaptive credit policies would not be able to recover it. Hence these results are a form of experimental validation of the previously derived theorems. Figure 5.5 shows three typical runs for different initial amounts of credit. A crunch is avoided in (a) via the activation of freeleech at several cycles - note the increase in credit over time. A crash is avoided in (c) via the activation of freeseed within the initial cycles - note the decreasing credit over time. In (b) a sustainable regime is shown in which the adaptive policy does not come into play and the credit over time remains constant.

5.4.2 Populations containing hoarder peers

In order to test whether the adaptive credit system can deal with more extreme conditions we ran simulations in which a small subset (1%) of the population are set as hoarders. As stated previously, any number of hoarders in a population will eventually lead to a crunch. This is because hoarders store-up increasing amounts of credit and eventually deprive all other peers of credit. Figure 5.6 shows two typical runs with and without the adaptive credit system. As can be seen, without adaptive credit the system eventually crunches, whereas the adaptive credit system keeps the system sustainable.

However, notice that the throughput of the system is very low because the adaptive credit system does not attempt to optimize the system but rather only to avoid a crunch condition. Hence the system injects new credit each time a crunch is predicted. This additional credit is eventually collected by the hoarders and the process repeats.

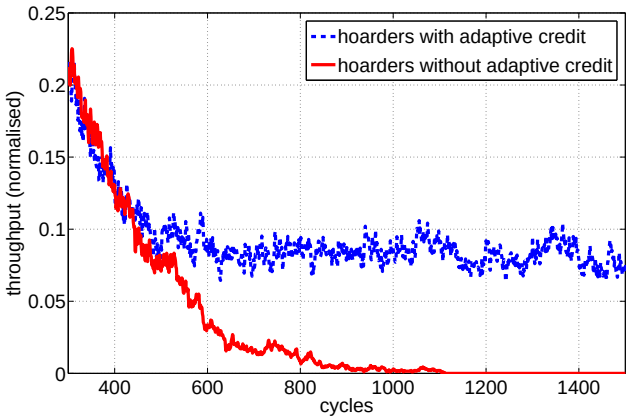


Figure 5.6: Two Test Runs (50% Rich Peers and 1% Hoarders).

We found that if we increase the number of hoarders to 5% of the population then the adaptive credit system could not prevent a crunch occurring. This is due to the user model assumptions we made in our analysis. That is, we assumed that peers will behave in the selfish way as described in Section 5.1.

5.4.3 Discussion

The aim of the adaptive credit system is to avoid crunch and crash states. This is achieved but comes at a potential cost. The reason for using credit systems within private communities is to provide incentives for peers seeding content. The freeleech and freeseed policies temporarily suspend these incentives. It could be argued that this could lead to reduced performance if users learned to game the system by only downloading during freeleech periods and not seeding during freeseed periods. However, as we have seen, the credit interventions only occur for short periods in our runs. This potentially means that it would be impractical for a user to notice and take advantage of such periods. An additional refinement, that could help preserve incentives even during freeseed and freeleech periods, would be to parameterize the freeseed and freeleech "tax" amount. This would mean that rather than always taking 100% of any leecher or seeder credit, other values such as 50% could be used. Any value less than 100% would still provide incentives for good behavior. Furthermore, the taxation amount could be variable, and could be applied in a continuous fashion, rather than getting triggered at the extreme conditions of crash and crunch.

As was observed in the situation where hoarders were introduced into the population,

the adaptive policy depends on the assumptions of the selfish user model since this formed the basis of our theoretical analysis. It was interesting to note that the system *could* cope with a small proportion of hoarders (following a different behavior from the selfish peers) but it is an open question as to how well it would cope with other small numbers of behavioral variants. We discuss this issue further in the conclusion section below.

5.5 Related Work

To the best of our knowledge, there has not been much work done on studying crunches, crashes and sustainability in P2P systems with credit based incentive schemes. In the previous chapter, we have introduced the concept of a credit crunch with the aid of a simplified model of a private tracker. We considered a very simple user model of all altruists and only concentrated on "limited" crunches in which insufficient initial credit in the system led to decreased (and not zero) throughput (Chapter 4). The present chapter builds upon that work and introduces a more nuanced user model, presents a theoretical analysis, and studies crunches and crashes that seize the entire system.

Many P2P incentive schemes based on credits have been proposed in the literature such as [36], [126]. These schemes usually build upon three components: 1) A virtual currency, 2) Micropayments, and 3) An accounting structure. In a P2P setting there are issues in maintaining these structures. For an accounting structure, such schemes usually have to rely on trusted accounting centers or third parties, although, the payments can occur in a decentralized way as proposed in [15]. Sirivianos et al. present monetary exchanges facilitated by a centralized bank [117]. Great emphasis is laid on creating a non manipulable scheme of exchanges using cryptographic techniques. The presence of a centralized bank means that the scheme is not scalable but has greater security than a completely decentralized solution.

Vishnumurthy et al. present a system involving virtual currency called Karma where sets of bank nodes keep the transaction balance of peers [128]. Karma is defined as the value which captures the amount of resources that a peer has contributed and consumed. This represents the users standing in the global system. Importantly, the level of karma (or credit) in the system is maintained and measures are taken to avoid inflation and deflation that can occur when peers leave the system. In this way [128] is an important contribution because the work begins to realize the problems that are inherent in dealing with credit systems. In avoiding inflation and deflation, their only aim is to maintain the per-capita karma i.e. the total karma divided by the number of active users.

Kash et al. [62] show that in a scrip system, where agents can consume and produce services, both an overabundance of money supply and its shortage lead to inefficiency. Surplus credit can lead to a monetary crash where freeriding is encouraged. At the other end of the spectrum, a shortage in money supply leads to agents not having enough money and not being able to afford services in the system. They also consider hoarders and how to optimize the credit supply. Our work is different, in that we focus not on a generic service exchange scenario but a filesharing scenario inspired by BitTorrent private communities. Also we apply a selfish user model[3] rather than a utility optimizing one. In addition, we focus on detecting and avoiding extreme crashes and crunches, where the entire system seizes, rather than optimizing the system.

The current deployed credit systems can be easily gamed since they rely on self-reporting of upload and download behavior by peers – currently such behaviour is policed by human administrators. Additionally the BitTorrent protocol itself can be gamed through strategic clients that act selfishly [68, 72, 93].

5.6 Conclusions

We have presented findings from an agent-based model of a private BitTorrent file sharing community which uses credits to incentivize uploading behavior. We have examined the credit dynamics found in the model with a population of selfish peers that only upload in order to continue to download. We identified, in simulation, conditions that lead to both crunches and crashes where the system completely seizes - meaning no further sharing activity is possible. We applied a theoretical analysis to precisely characterize these conditions that lead to the system crunching or crashing. We validated the analysis against simulation runs and applied the derived conditions to implement a novel *credit intervention mechanism* that proactively stops the system seizing by temporarily changing the credit policies. A system that is predicted to crunch allows freeleeching (i.e. downloaders do not use any credit but uploaders still gain credit) conversely a system that is predicted to crash imposes freeseeding (i.e. downloaders use credit as normal but seeders do not gain credit). These interventions are only applied while the system is in such critical states. Freeleeching injects credit into the system and freeseeding removes credit from the system.

Our findings are based on the assumption of a selfish user model and an abstracted private community model. We have purposefully excluded differential upload and download

[3]The selfish user model could be interpreted as a so-called "satisficing" model since peers stop trying to gain credit when they reach a satisfaction threshold.

rates and other plausible user model variants in order to understand the effect of credit dynamics in a system with known properties. Even given our assumptions we found that understanding and predicting credit dynamics and system behaviour was far from trivial.

Given these issues we can not claim that our adaptive credit policy could currently be deployed in a real private tracker community but would almost certainly need to be refined and informed by empirical work.

Possible future work could involve introducing a distribution of user model variants to our simulation model and analysis using a probabilistic rather than a deterministic approach. This may allow for some level of probabilistic prediction of crash or crunch states. More importantly, we may be able to induce probabilistic user models from empirical data collected from real private communities.

Chapter 6

Improving Efficiency and Fairness in P2P Systems with Effort Based Incentives

Traditional methods of rewarding peers in P2P systems based on contribution (total bandwidth offered) lead to the welfare of the fast peers and disfavor slow peers. As discussed in Chapter 2, it has generally been argued that peers that contribute more should receive better service than those that contribute less [25, 68].

In this chapter, we apply a new principle of reward to P2P systems, which has been inspired by Participatory Economics (*Parecon*). Parecon is an alternative economic vision in which reward is based on relative contribution or *effort*, defined as the contribution as a fraction of capacity, instead of (the absolute value of) the contribution itself [2]. We argue that this principle (rewarding according to effort instead of contribution) can be adopted to design P2P systems that are efficient while being fair.

Rewarding according to effort, while being incentive compatible, gives both slow and fast peers in the system a level playing field. All peers, regardless of their bandwidth, *can potentially* make the same level of effort. On the other hand, slow peers simply cannot compete with fast peers in terms of output or contribution levels. We argue that systems are fair if they reward effort instead of output and hence are equitable to the less resourceful peers.

P2P systems provide plenty of opportunities for peers to undertake actions that can be characterized as effort. It is a reasonable aim to get peers to share content for long periods of time at as high rates *as they possibly can*. So, their *relative* contribution is a good approximation of the effort they make for the welfare of the community. Possibly, other forms of effort such as sharing rare content, taking the time to rate content, and helping other peers communicate through NATs and firewalls, can also be taken into account. In this chapter, we only consider effort in terms of bandwidth offered relative to capacity.

In order to validate our approach, we performed a range of experiments in which we apply the principle of *reward according to effort* to BitTorrent. We show that not only does our adaptation lead to more fairness, but even to a higher average system performance. Specifically, in the presence of a high proportion of fast peers, we observe the download speed of slow peers to increase up to 63% at only a marginal 4% decrease in speed for fast peers. Also, in the unmodified BitTorrent protocol, fast peers can achieve as much as 60% higher speeds than slow peers, whereas with our policy, the speeds of the two groups converge to almost identical values, with fast peers reaching speeds only 2% higher than slow peers.

To verify that our approach can be applied to a variety of systems, we also applied our effort-based reward approach to a generalized credit based sharing ratio enforcement scheme, and show how it positively affects performance and fairness. We observe that rewarding according to effort more than doubles the average download performance of the system. At the same time, the inequality in the system decreases by approximately 80%. Overall, we present an alternative approach that inspires the design of incentive mechanisms for P2P systems which are better suited to the heterogeneous nature of the Internet and its users.

In Section 6.1, we discuss how the notions of efficiency, fairness, social welfare, and incentives, have been utilized in the literature and implicitly in P2P systems. In Section 6.2.1, we present results from applying effort based rewards to BitTorrent and compare it with the existing protocol. In Section 6.2.2, we present results from applying effort based rewards to a generalized credit based sharing ratio scheme. We conclude the chapter in Section 6.3, with a summary of our approach and results.

6.1 Efficiency, Fairness and Incentives

In this section we discuss how incentives have been utilized, and the notions of efficiency and fairness formulated, in the literature and implicitly in P2P systems.

What do system designers want from P2P systems? What kind of economic incentives do they desire? Usually, a combination of the following goals has been sought:

- *More cooperation and less selfishness*
- *More efficiency and less wastefulness*
- *More equity and less unfairness*

We shall next consider each of the desired goals in turn.

6.1.1 More Cooperation and Less Selfishness

Fostering cooperation and eliminating selfishness is the primary goal of all incentive-based systems in P2P. We want to provide incentives to peers to contribute their resources to the network. Resources could encompass content, time spent sharing content, and the rate at which content is contributed.

6.1.2 More Efficiency and Less Wastefulness

What do we mean by more efficiency? Normally Pareto optimality has been employed by P2P designers to measure efficiency. The first paper on BitTorrent, highlights the achievement of obtaining Pareto efficiency [18]. Indeed in an influential paper, Dash et al. in detailing the desiderata of mechanism design, list Pareto optimality as one of the sought after goals [21]. It is worthwhile to study what exactly Pareto optimality entails.

Pareto Optimality: A change from one allocation to another that can make at least one individual better off without making any other individual worse off is called a Pareto improvement. An allocation is Pareto efficient or Pareto optimal when no Pareto improvement is possible [132]. Pareto optimality is not necessarily fair [111]. For example, allocating all resources to one individual and giving nothing to the rest is also a Pareto optimal solution.

What is striking is that if P2P designers were to adhere strictly to Pareto optimality, then they would not have much, if anything, left to propose. This is because most solutions make *some* peers better off and *some* peers worse off. This has implications for designers working on improving existing protocols. It should be remembered that most recommendations for changes in policies, such as modifying BitTorrent's unchoke policy, are not Pareto improvements since they make some peers worse off.

Mainstream economists try to circumvent this problem by using an extended concept of efficiency called the *efficiency criterion*, which says that if the overall benefits to any and all people of doing something outweigh the overall costs to any and all people, it is efficient to do it, and vice versa. As in mainstream economics, in P2P as well, social welfare has been equated with efficient outcomes [5, 21, 74]. Next, we shall discuss why this equation of social welfare with efficiency is inadequate.

6.1.3 More Equity and Less Unfairness

How can it be decided that the overall benefits to some people outweigh the costs to some other people? In the context of P2P file sharing networks, who is to say that it is efficient to provide reduced download times to faster peers while increasing the download times of slow peers? Fact of the matter is that *value judgements* are implicit in the efficiency criterion. A designer has to make value judgements on what she/he feels is a better solution.

One designer might decide that increasing the utility of individual peers is of foremost importance while another designer might decide that the chief aim should be to increase the average system performance.

The point is that the principles and values we follow, dictate how we formulate and answer such questions. Based on personal judgement, a designer has to attach weights to the well-being of different peers.

We desire efficient outcomes, but such that they are fair and equitable to the less resourceful peers in the system. We therefore argue that the stated goal of numerous incentive works in P2P (e.g., in [21]: "achieving efficient outcomes for social welfare"), is inadequate since it is not qualified by a condition of equity.

It could be argued that it is fair that the fast peers, who contribute more to the system in terms of volume, overall get better service and more rewards from the system. However, this definition of fairness assumes maxims of remuneration that reward peers for their fast connections. There are two familiar maxims of remuneration [40]:

a) *Payment according to value of one's personal contribution and contribution of the productive property one owns.* Peers should get out of the economy what they and their productive property (reputation or virtual money in case of P2P) contribute to the economy.

b) *Payment according to the value of one's personal contribution only.* Peers should get out of the economy how much they contribute to the economy. This is in fact how peers are currently remunerated in P2P settings. The faster the connection a peer has, the faster it will be able to download (BitTorrent) and the more currency or reputation it will be able to earn in monetary or reputation based schemes.

It is clear that maxims *a)* and *b)* favor the faster peers who will be rewarded higher in a system that utilizes either of these two maxims.

We now consider an economic system that utilizes a novel, third, maxim of remuneration, which in our view can facilitate the achievement of efficiency with equity.

6.1.4 Participatory Economics

In view of our goal of achieving efficiency with equity, we turn to Participatory Economics (*Parecon*), an alternate economic vision developed by Michael Albert and Robin Hahnel [2]. A comprehensive overview of *Parecon* is available at [138]. For the purposes of this study, we shall concentrate on the *Parecon* principle of remuneration:

Payment according to effort and sacrifice. This maxim suggests that people should be rewarded for the efforts and sacrifice that they put into their work, rather than being paid for their output.

We argue that for P2P systems, utilizing this principle can be used to design incentive schemes that are fair, because it ensures that peers who do their best to contribute to the

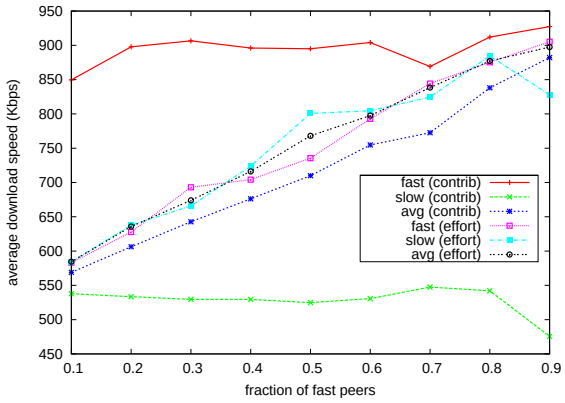


Figure 6.1: The download speed of both fast and slow peers in networks with various fractions of fast peers, for the contribution-based policy (contrib) and the effort-based policy (effort) in BitTorrent.

system are rewarded even though they might not be well endowed in terms of bandwidth.

6.2 Efficiency and Fairness in Deployed Mechanisms

In order to assess the efficiency and fairness of effort-based incentive schemes in practice, we applied our approach to two currently deployed mechanisms: the BitTorrent protocol and a generalized credit based sharing ratio enforcement scheme. We observed promising results, with our approach yielding higher efficiency and fairness for both.

6.2.1 Efficiency and Fairness in BitTorrent-like Systems

We simulated systems using both the original BitTorrent policy (contrib) and our Parecon policy (effort). We used a BitTorrent simulator that accurately mimics the behavior of BitTorrent at the level of individual piece transfers, based on BitTorrent's *unchoke policy* and *rarest-first piece selection* [18]. In the original BitTorrent policy, peers reciprocate according to the received contribution from others; in the Parecon policy peers reciprocate according to the effort another peer is giving, defined as the bandwidth it is giving relative to its upload capacity. To be more precise, a peer i periodically decides to whom it will allocate its upload slots by ranking the other peers according to values r_j where for a peer j it holds that: (i) $r_j = b_{ji}$ in the contrib policy; (ii) $r_j = b_{ji}/U_j$ in the effort policy. Here b_{ji} is the amount of bytes uploaded by peer j to peer i in some sliding window of time, and U_j the upload capacity of peer j. (We note that a secure, efficient, and accurate mechanism to determine the capacities of nodes in a P2P network has been published recently [118].) In addition, in the BitTorrent protocol a peer periodically allocates a slot to a random peer, which we left unchanged.

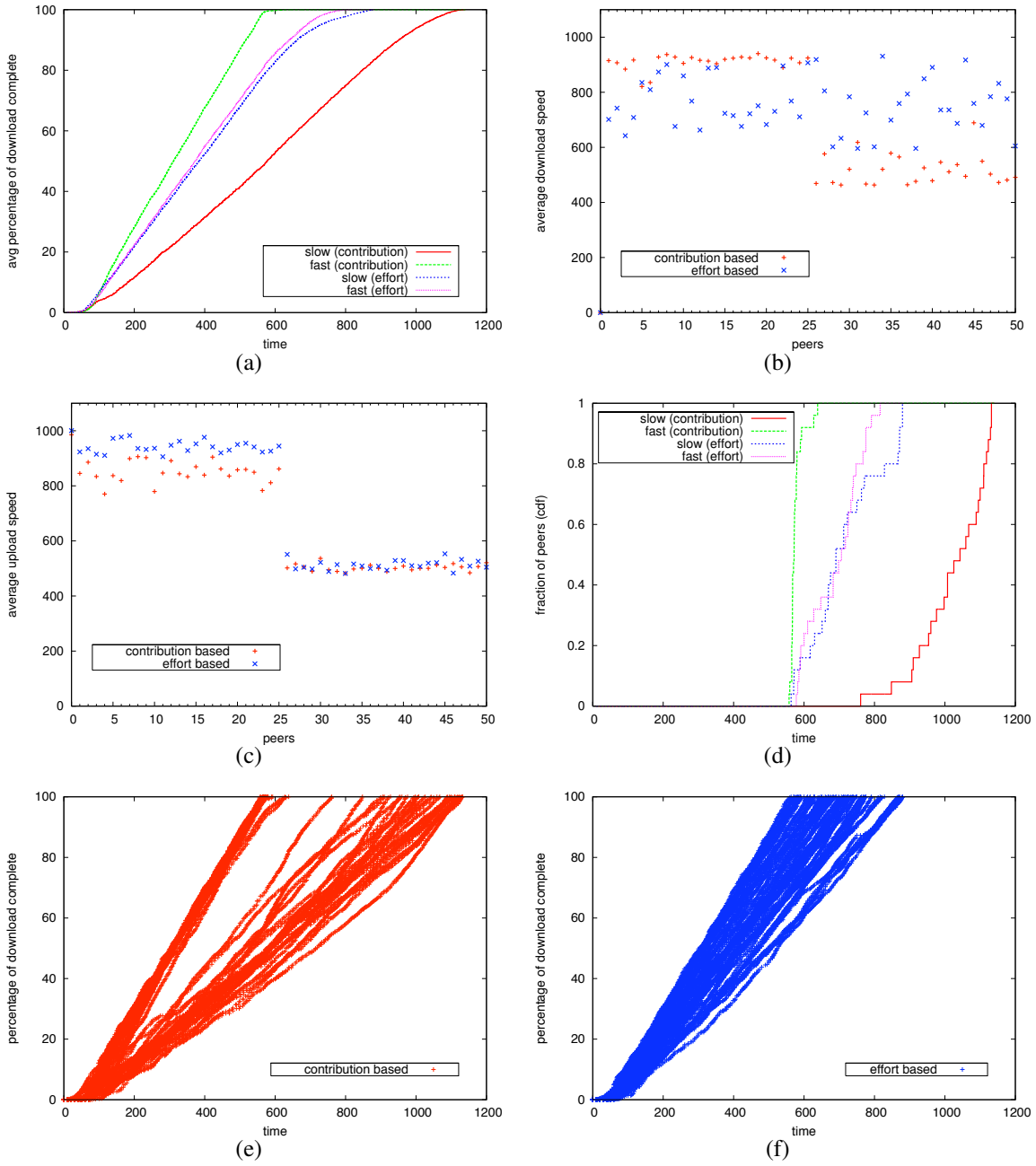


Figure 6.2: The performance of the contrib and effort policies in BitTorrent with 50% fast peers and 50% slow peers.

We have simulated systems with swarms of 50 peers consisting of two classes: fast peers with an upload capacity of 1024 Kbps and slow peers with an upload capacity of 512 Kbps, in varying proportions. This polarized view allows us to clearly analyze the effect of a peer's capacity on its performance. Furthermore, we assume that the download

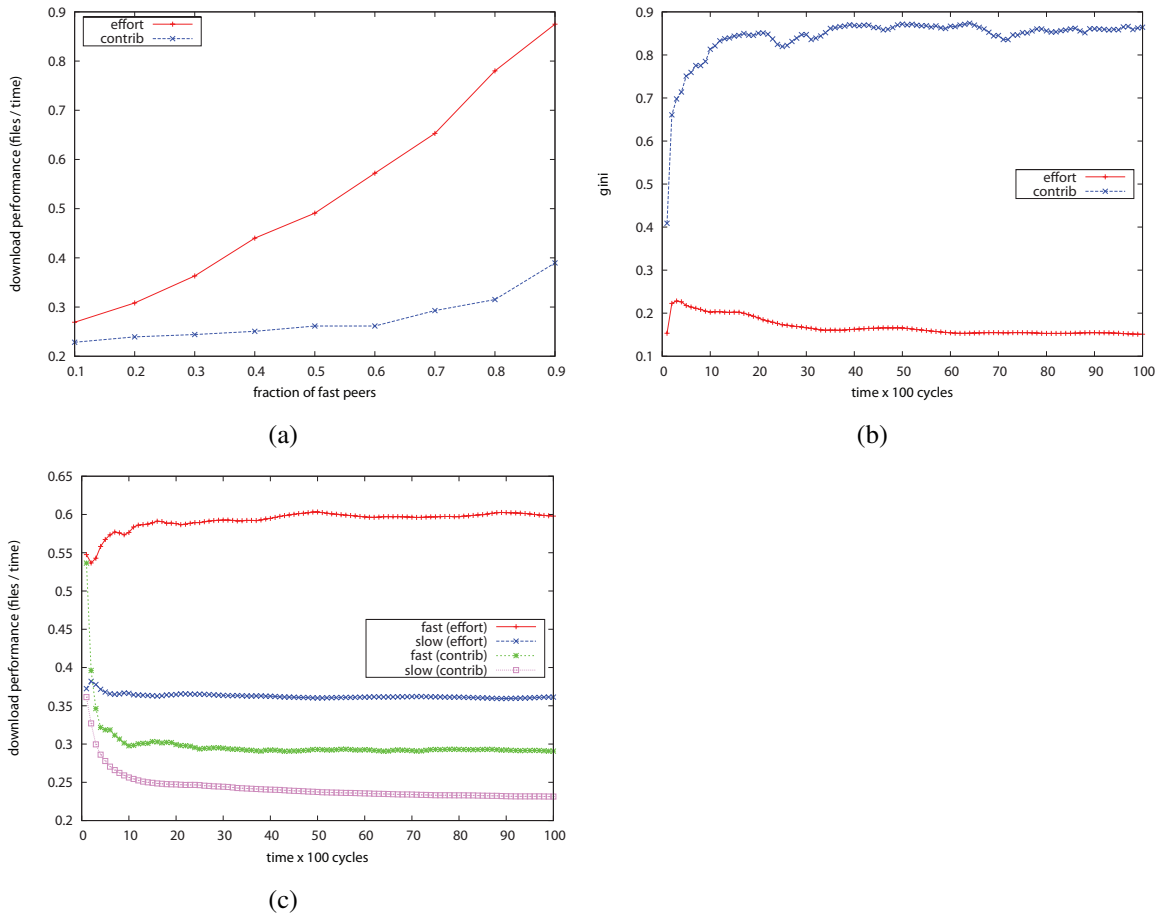


Figure 6.3: The performance of contrib and effort policies in Credit Based Scheme.

bandwidth is not a bottleneck, which corresponds to most realistic BitTorrent systems as most users have highly asymmetric bandwidth capacities [110].

In Fig. 6.1, the download speed of both classes of peers (averaged over all peers in that class) is displayed for systems with various fractions of fast peers. To our surprise, we observe that under all configurations the average download speed under the effort policy is *higher* than under the contrib policy. The effort policy is not only more fair, but also leads to an overall faster distribution of content, thereby dismissing classical claims that contribution-based reciprocation is necessary to optimize overall system performance. When there is a high fraction of fast peers, the download speed of slow peers can increase up to 63% at only a slight loss for fast peers: a 4% reduction in speed. On the whole, the average system download speed is higher under all configurations and can increase by as much 10% (Fig. 6.1).

As expected, effort-based reciprocation is much more fair: compared to the polarized speeds observed with the contrib policy, the effort policy treats slow and fast peers much more evenly. In the unmodified BitTorrent protocol, i.e. the contrib policy, the fast peers

can achieve as much as a 60% higher speed than slow peers whereas with the effort policy, the speeds of the two groups, converge to almost identical values, with fast peers reaching speeds only 2% more than slow peers (Fig. 6.1).

When there are only a few fast peers, these few have to sacrifice a lot in the effort policy against a meager improvement for the slow peers; when there are many fast peers, each has to sacrifice only little while there is a huge improvement for the slow peers. Fig. 6.2 show the properties of both policies in more detail for a system with 50% fast peers and 50% slow peers. Fig. 6.2(a) shows the download progress over time; Fig. 6.2(b) shows the average download speed of all 50 peers in decreasing order of capacity; Fig. 6.2(c) shows the average upload speed of all peers; Fig. 6.2(d) shows cumulative distribution function of the download finish time; Fig. 6.2(e) shows download progress for all peers using the contrib policy; Fig. 6.2(f) shows download progress for all peers using the effort policy. The average download completion times of fast peers is nearly 50% shorter than slow peers in the contrib policy, while this difference comes down to almost 5% in the effort policy (Fig. 6.2a). Importantly, under the effort based policy, an increase of up to 25% in the average upload speed is observed as well (Fig. 6.2c). Fig. 6.2d shows the cumulative distribution function of the download finish time. Fig. 6.2e shows the download progress for all peers using the contrib policy. And Fig. 6.2f shows the download progress of all peers using the effort policy. Overall, the effort policy leads to higher average upload speed, shorter download times, and a smaller variation in finishing times of slow peers. The first two points substantiate the claim that the effort based policy leads to a more efficient system with greater utilization of available resources and increased overall system performance while the last point demonstrates the fairness of the policy.

Hence, overall these experiments show that an effort-based policy in BitTorrent is advantageous regarding both system efficiency and fairness. The only subjective disadvantage is that the fastest peers have to 'sacrifice' some of their performance to the benefit of others, which we would argue is a very reasonable property of P2P systems both from a designer and user point of view[1].

6.2.2 Efficiency and Fairness in Credit Based Enforcement Schemes

In order to establish that our approach can be applied to a variety of systems, we applied an effort based incentive policy to a simplified version of a credit based sharing ratio enforcement scheme. Sharing ratio enforcement schemes are used by most private BitTorrent sites, called trackers, in order to incentivize sharing (seeding) by peers. In such schemes, a peer is only allowed to download as much as it uploads. A peer can build a positive ratio or earn credits by seeding content to other peers. Sharing ratio enforcement,

[1] This can be compared to redistribution mechanisms to promote social welfare in human societies.

then, is an incentive mechanism for peers to seed. Seeding is done by peers who stay voluntarily in the network after finishing their own downloads.

If a peer i uploads to peer j, the former's account is credited while the latter's is debited. A peer can only continue downloading if it has a positive credit (above zero). Thus the total credit in the system will be an invariant C whose distribution over peers will vary with time. In such a scenario, peers with low upload speeds would naturally accumulate less credit as compared to fast peers if both seeded for the same amount of time. *It should be noted that such credit based sharing ratio schemes are independent of the BitTorrent protocol. Hence, the efficiency and fairness of such systems is solely dependent on credit dynamics.*

We argue that such systems are ideal candidates for the application of effort-based incentive schemes. If all peers are given credit for the *time* they seed content rather than the amount in megabytes, then all peers could potentially come on a par.

In order to test our hypothesis, we performed experiments using a simplified model of a credit based sharing ratio enforcement scheme. This model is similar to the one described in our previous work in Chapter 4. In this model, the community is represented by a set of peers (P). All peers have fixed upload capacities. We employ a very simple user model; All peers are online at all times. At any given time, a peer is seeding some number of swarms (S) and downloading from some number of other swarms (D). Peers seed files for some fixed amount of time and then remove them from their seeding list. In our experiments we set $D = S = 1$ and the maximum seeding time set to infinity. This means that each peer is always downloading in one swarm and seeding in one other swarm.

We ran the simulations for various file sizes and various speeds for fast and slow peers. We found that these have no effect on the results. This is because all peers are downloading in one swarm and uploading in another at all times, and the only factor that can affect the difference in efficiency of the fast and slow peers, and credit inequality, is the relative difference in the speeds of fast and slow peers.

We ran the simulations on two policies: contribution based contrib and effort based effort. In contribution based policy, each peer earns credit based on its upload speed. So if a peer has an upload speed of x units per time unit, then it earns x units if it seeds a file for one time unit. In the effort based policy, all peers earn the same credit for seeding for one time unit, regardless of their upload capability.

We found that the effort based policy not only leads to a fairer system (lower Gini)[2] but also to a more efficient system in which the overall download performance of all peers increases. (Here we have defined download performance as the number of files downloaded by a peer per one time unit.) Fig. 6.3a and 6.3b show that the average system efficiency and fairness increase when the effort based policy is applied. Specifically,

[2]The Gini coefficient is a number ranging from 0 to 1 that characterizes inequality with 1 being the most unequal (one peer holds all credit) and 0 being complete equality [131].

rewarding according to effort increased the average download performance of the system by more than 100%. Also, the system becomes much fairer with an approximate 80% decrease in credit inequality.

Fig. 6.3c shows the somewhat surprising result that the performance of *both* fast and slow peers goes up under the effort based policy. This appears to be counter-intuitive. However, it can be explained by the fact that rewarding peers according to effort results in an injection of new credits in the system and as we showed in Chapter 4, injecting new credits in the system leads to a more efficient system. This is due to the fact that because of extra credit, slow peers are not 'strapped for cash' so to speak, simply because they are slow, and thus are able to download more files, increasing overall system performance.

6.3 Conclusion

In this chapter, we explored the use of incentive mechanisms in P2P systems. We argued that rewarding according to effort instead of contribution would lead to systems that are efficient and fair. This principle has been borrowed from Participatory Economics (*Parecon*), which is an alternate economic vision.

We presented simulation results of applying our approach to currently deployed mechanisms. We modified the popular file sharing protocol BitTorrent to reward according to effort. Upon doing so, we made the surprising discovery that rewarding according to effort rather than contribution, makes BitTorrent not only much fairer but also more efficient. We observed up to 60% increase in the download speed of slow peers and up to 10% increase in the average system download speed. The huge difference in the download speeds and download completion times between slow and fast peers was also greatly reduced.

We also applied our approach to a credit based ratio enforcement scheme. Here too, we noticed that rewarding according to effort makes the system both fairer and more efficient. The average download performance of the system more than *doubled* and there was approximately 80% decrease in credit inequality.

In the future, we want to test our approach and analyze its feasibility in the presence of freeriders, who are determined to make no effort. Also, we intend to borrow other principles from Parecon for improving the design of P2P systems.

Chapter 7

Design Space Analysis for Modeling Incentives in Distributed Systems

Our major observation in this thesis can be described as the importance of incentives in distributed systems with no centralized authority. Robust incentives ensure that the prescribed protocol is followed by all the nodes, i.e., the protocol is robust to strategic manipulation. A powerful tool for modeling incentives is game theory, the branch of economics that can model individual behavior in strategic situations [90]. The general applicability and predictive powers of game theory has allowed designers to employ it in a variety of contexts for the design of distributed systems [14, 106, 112]. However, little attention has been paid to exploring new approaches that can be equally generally applied, and which can complement game-theoretic analysis. The reader will remember that we discussed this point earlier in Chapter 2.

Solution concepts from game theory often require high levels of abstraction to keep the models simple. This is required because involved models can become analytically intractable [76]. Thus, for modeling complex protocols, a game-theoretic analysis runs the risk of missing out on important variables that could have significant effects on protocol design. For instance, while designing a protocol, it is not uncommon for the designer to employ a variety of arbitrary design decisions and so-called "magic numbers". Modifying any of these can have negative effects on the robustness of the protocol's incentives. In other words, there are many elements in complex protocols that could be gamed by strategic nodes.

In this chapter, we aim to devise a method which can be used to analyze protocols more comprehensively. To that end, we present a simulation-based approach that we call *Design Space Analysis* (DSA). DSA combines the specification of a design space with an analysis of varying protocols within that space[1].

[1]The experiments for this chapter were performed on the *Distributed ASCI Supercomputer 3* (http://www.cs.vu.nl/das3/)

The specification of a design space comprises two steps: *Parameterization* and *Actualization*. *Parameterization* involves identifying salient design dimensions for the space, while *Actualization* involves specifying multiple implementations for the identified dimensions.

For an analysis of the design space, we present a solution concept inspired by the work of Axelrod [7], which we term the *Performance, Robustness*, and *Aggressiveness* (PRA) quantification. For a protocol Π, *Performance* is the overall performance of the system when all nodes execute Π (where performance is defined by the application); *Robustness* is the ability of a majority of the population executing Π to outperform a minority executing a protocol other than Π; and *Aggressiveness* is the ability of a minority of the population executing Π to outperform a majority executing a protocol other than Π. PRA quantification takes the form of a tournament in which each protocol competes against every other protocol. By evaluating each protocol in the space, the PRA quantification simulates strategic variants and predicts their effects.

The two approaches for protocol analysis and design are depicted in Figure 7.1. In the rest of this chapter, we examine the complementary nature of the two approaches. We achieve two aims by exploring both approaches: 1) We demonstrate that for game-theoretic analysis of complex protocols, the abstractions that designers choose determine the result that they obtain; different abstractions can lead to different, even contradictory, results. Hence, the results obtained from a game-theoretic analysis, while valuable for being powerful predictors obtained from simple abstractions, need to be analyzed in more detail; and 2) We establish DSA as a viable and tractable tool for a detailed analysis of protocols. The results of DSA can be used to gain insights into the workings of protocols and for designing deployable systems.

We explore both approaches by applying them to peer-to-peer (P2P) systems. We choose P2P systems because there are many such deployed systems, for which incentive-compatible design is of primary importance to counter strategic behavior. We undertake the following steps. We apply a game-theoretic analysis to the popular P2P protocol, BitTorrent, and devise a more robust variant (Section 7.1). In order to thoroughly analyze this variant we perform a Design Space Analysis of P2P file swarming systems. We run experiments on a cluster and discover that there are several protocols that do better than this variant with respect to Performance, Robustness, and Aggressiveness (Section 7.2 & 7.3). Finally, we implement modifications to BitTorrent and with experiments on a cluster, analyze some protocols discovered using DSA. We show that they yield higher system performance and robustness as compared to the reference implementation, thus demonstrating the effectiveness of DSA (Section 7.4).

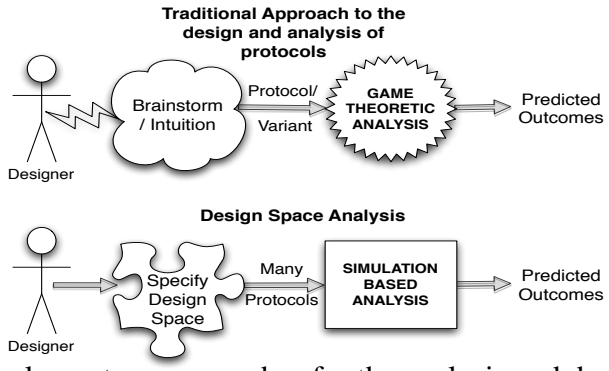


Figure 7.1: Complementary approaches for the analysis and design of protocols.

7.1 Game-Theoretic Analysis of BitTorrent

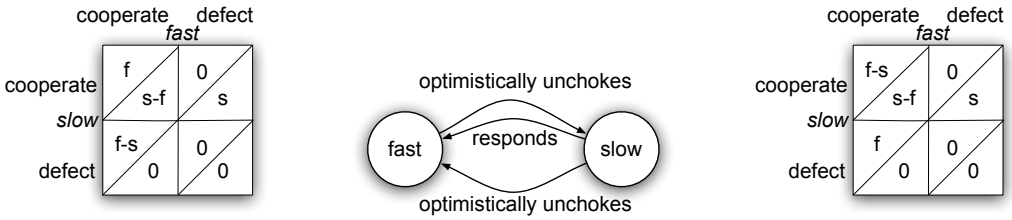
We consider one of the most popular P2P protocols, BitTorrent (BT), for our analysis. Our reason for choosing BitTorrent is that this protocol has probably been the most widely studied P2P protocol in the literature. The 'Traditional Approach' depicted in Figure 7.1 has often been followed for BitTorrent: a protocol improvement or variant is designed, and then the variant is either subjected to a game-theoretic analysis and/or shown to be better than some chosen protocol(s) in simulations [10, 60, 68, 94].

First, we present a model of BitTorrent as a *strategy* in a *game*. In game theory, a game is a description of a strategic interaction that includes the constraints on the actions that the players can take and the players' interests [90]. Then we present an analysis of this model for multiple bandwidth classes. Under our assumptions, which are different from previous work [97], we show that BitTorrent is not a Nash equilibrium. Finally, we design a modification to BitTorrent, which is a Nash equilibrium.

We assume the reader is familiar with certain game-theoretic constructs such as the Prisoner's Dilemma (PD), a game between two players in which it is the dominant strategy of both players to defect.

7.1.1 BitTorrent as a strategy in a game

We explain the basics of the BitTorrent protocol from an iterated games setting perspective. Each peer plays a number of games with other peers in a given time period, following a Tit-for-Tat (TFT) like strategy. TFT is the strategy using which a player *cooperates* on the first move and then simply mimics what the other player did in the last round. In BitTorrent a peer cooperates with (i.e., uploads to) a certain number of preferred (fastest uploading) partners while it defects in the rest of the games. These are the 'regular unchokes' in BT terminology. Additionally, a peer also starts new games with other peers in search of better partners. These are 'optimistic unchokes' in BT terminology. In these games, a peer always cooperates unconditionally for some iterations. We do not model



(a) (b) (c)

Figure 7.2: Analysis of the BitTorrent Dilemma: (a) The payoffs of the BT Dilemma for slow and fast peers; (b) An abstract illustration of interaction between slow and fast peers; (c) Modified BT payoffs in view of slow peers' opportunity costs.

the seeders in BitTorrent as these do not affect our subsequent Nash equilibrium analysis.

We now present an analysis of our model in a system containing two classes of peers: fast and slow. The game interaction in Figure 7.2(a) captures the dynamics between a fast peer and a slow peer, where f is the upload speed of a fast peer and s is the upload speed of a slow peer. This game represents a single round in an iterated scenario, where the 'shadow of the future' is large (i.e., the payoff of subsequent moves is important relative to the previous move) and peers can form sustained relationships. It can be seen that given the payoffs, the dominant strategy for fast peers is to always defect on the slow peers. This is because when a fast peer cooperates with a slow peer, there is an *opportunity cost* associated with it. Opportunity cost is an important concept in economics. It is the cost of an alternative that must be given up in order to pursue a certain action [41].

A fast peer's opportunity cost in cooperating with a slow peer is a missed interaction with another fast peer. When a fast peer cooperates with a slow peer, it gets a negative utility of $s-f$. It gets s from the slow peer but on the other hand, loses out on a potential f from a fast peer. Conversely, for the slow peers, the dominant strategy is to always cooperate with the fast peers. A slow peer on defecting against a fast peer gets f from the fast peer and can form a relationship with a slow peer, where it gets $s-f$ (where $-f$ is the opportunity cost of cooperating with a slow peer), thus getting a final utility of $f+(s-f)=s$. Figure 7.2(b) depicts this scenario: a slow peer responds (with cooperation in the form of a 'regular unchoke' slot) upon being optimistically unchoked by a fast peer, while the converse does not hold. In light of this, we note here that the Prisoner's Dilemma is not an accurate model for BitTorrent under heterogeneous classes of peers. Instead, the way BitTorrent implements the interaction of a slow peer with a fast peer, resembles an interaction in the *Dictator game*, a game in which one player proposes to do something, while the other has no choice but to respond passively without any strategic input into

the decision. It also resembles a game which has been called by some as the *One-Sided Prisoner's Dilemma* [105]. For simplicity, we refer to it here as the *BitTorrent Dilemma*.

We note here that Levin *et al.* [68] have suggested that BitTorrent can be modeled as an *auction*. Using an auction model they were able to design a more robust protocol variant. However, by considering our 'game' abstraction, we have unearthed new aspects of the protocol and will also present a more robust variant in Section 7.1.3. Thus we highlight that when applying game-theoretic analysis, there is no single *right* abstraction. Even when the analysis is performed on the same point in the design space (peer selection in this case), different designers can choose different abstractions to yield different insights. Hence, seemingly conflicting statements such as *BitTorrent is an auction* or *BitTorrent is a strategy in a game* can be said to be equally valid. We observe this point again, more conclusively, when we perform a Nash analysis for BitTorrent in Section 7.1.3 [2].

Next we give an analytical model of BitTorrent for multiple bandwidth classes, using the BitTorrent Dilemma game as depicted in Figure 7.2(a).

7.1.2 Analytical model of BitTorrent Dilemma

In this section, we model the BitTorrent Dilemma game with multiple bandwidth classes of peers. We seek to calculate the expected number of games that a peer c from a particular class, with payoffs defined according to Figure 7.2(a), can win against other peers, where winning means getting cooperation from others.

In the remainder of this section we derive the formulae for the expected number of games that peer c wins against other players from different classes. We note that there are two types of games that a player c can win: 1) the games that it wins when others *reciprocate* to it; and 2) when other players start a new game with c and in line with TFT, cooperate unconditionally, thereby giving c a *free game win*.

We use the notation summarized in Table 7.1. Note that we assume, for notational simplicity, that the number of new partners that a peer cooperates with unconditionally (number of optimistic unchoke slots in BT terminology) is equal to 1. Moreover, it is also assumed that there are always enough peers to exchange a particular piece of content. The basic interactions between the different classes of peers are shown in Figure 7.3.

First, we calculate the expected number of games that can be won against higher classes. We assume that N_A is greater than U_r; thus, as per Figure 7.2(a), players employing TFT in higher classes will not reciprocate to peer c. Therefore, $E^r[A \rightarrow c] = 0$.

However, as per the TFT policy, peers from higher classes, unknowingly, do offer first move cooperation to peers from lower classes in search for better partners. The probability that peers in class C are offered a 'free game win' by a peer from the higher

[2] For a more general discussion on whether BitTorrent can aptly be described as using TFT, we refer the reader to Appendix B

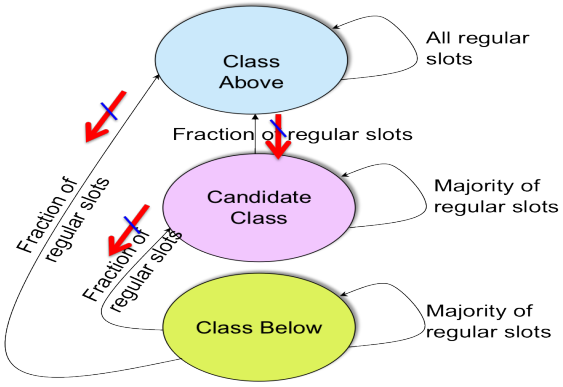


Figure 7.3: An N Class Analysis of the BitTorrent Dilemma.

Table 7.1: Model parameters. Classes are based on peers' bandwidth capacities.

Notation	Definition
N_A	number of TFT players in classes above c's class.
N_B	number of TFT players in classes below c's class.
N_C	number of TFT players in c's class.
U_r	number of players that c can reciprocate with simultaneously (number of regular unchoke slots in BT)
$E^r[X \to c]$	the expected number of games peer c wins against peers in the class $X \in \{A,B,C\}$, where A,B and C are the classes above, below and where peer c is from, respectively.
$E[X \to c]$	the expected number of 'free game wins' that peer c obtains from class $X \in \{A,B,C\}$.
N_r	$N_A + N_B + N_C - U_r - 1$

classes is N_C/N_r, giving $E[A \to C] = N_A \times N_C/N_r$, which is the expected number of 'free game wins' that peers from higher classes offer to players in the considered class C. This leads to $E[A \to c] = N_A/N_r$.

For the expected number of games won against the lower classes, using similar reasoning as above, we obtain $E[B \to c] = E^r[B \to c] = N_B/N_r$.

The expected value for the number of games in which a peer c gets reciprocation from peers in the same class is U_r *minus* the number of 'free game wins' that peer c obtains from the higher classes (to which c always reciprocates as per its dominant strategy, thus breaking its relationship with a partner from the same class), *minus* the expected number of 'free game wins' that at least one of c's current partners gets from the higher classes

(to which the partner reciprocates, thus breaking its relationship with c). This leads to

$$E^r[C \to c] = U_r - E[A \to c] - K, \tag{7.1}$$

where $K = 1 - \big((1 - E[A \to c])(1 - \frac{1}{U_r})\big)^{U_r}$. Finally, the number of peers in contention for 'free game wins' by peer c in the same class is $N_C - 1 - E^r[C \to c]$, which gives $E[C \to c] = (N_C - 1 - E^r[C \to c])/N_r$. (We note that K is a conservative estimate, the value of which can be greater if more than one of a peer's partners decide to end reciprocation simultaneously. However, a greater value of K in Eq. 7.1 would only serve to strengthen our subsequent Nash analysis and so we can comfortably ignore it.)

7.1.3 Is BitTorrent TFT a Nash equilibrium?

Under certain assumptions, it was shown in [97] that the TFT strategy as implemented in BitTorrent is a Nash equilibrium. However, by modifying the abstraction to incorporate more detail in our model, and taking cue from formula (7.1), we find that BT is not a Nash equilibrium. In our model, the payoff structure in BitTorrent can be modified to devise a new protocol. This protocol can individually, in a population of all BitTorrent peers, do better than the BitTorrent peers, thereby showing that BitTorrent is not a Nash equilibrium.

Next, we discuss how we devise a protocol improvement called *Birds*[3] by modifying the payoffs in the BitTorrent Dilemma.

Birds: Modifying BitTorrent's payoffs. A fast peer upon being optimistically unchoked by a slow peer does not reciprocate, as given by Figure 7.2(a), because a fast peer realizes the opportunity cost of reciprocating to a slow peer, which is a missed interaction with another fast peer. A slow peer reciprocates to a fast peer because in its view there is an opportunity cost in defecting against a fast peer, which is a missed chance to form a long-term relationship with a fast peer. However, as stated, given the scenario described in Figure 7.2(a), this does not happen. This suggests that the payoff as calculated by a slow peer in the BitTorrent Dilemma could be modified. There is no opportunity cost in defecting against a fast peer. In fact there *is* an opportunity cost in cooperating with it: missing out on a sustained relationship with another slow peer. Therefore, in order to account for this fact, we modify the payoff structure of the BitTorrent Dilemma according to Figure 7.2(c) so that the dominant strategy of both slow and fast peers is to defect against each other.

Analytically, this leads to $E^r_B[A \to c] = E^r_B[B \to c] = 0$, and obviously $E^r_B[C \to c] = U_r$. For the 'free game wins', there is no change as compared to BitTorrent. For the

[3]Birds of a feather stick together. Using this protocol peers try to stick with others from their own bandwidth class.

interactions of peers in the same class, we find that we can use the same argument as used for BitTorrent, thus $E_B[C \to c] = (N_C - 1 - U_r)/N_r$.

For a proof of Birds being a Nash equilibrium and BitTorrent not being a Nash equilibrium, we refer the reader to Appendix A.

A practical approach to deploy Birds. We provide a simple, practical approach to incorporate Birds in current BitTorrent clients. We propose to change the peer selection policy of BitTorrent, so that it reciprocates not to the fastest peers but to those peers that are closest to its own upload bandwidth. Given Figure 7.2(c), a peer in Birds needs to sort others in increasing order of their *distance* to its own upload bandwidth.

We define a peer's distance to another peer to be equal to the absolute difference between its upload speed and the upload speed of the other peer. We note that a similar approach has been used for replica placement in P2P Storage Systems [107].

7.1.4 Discussion

We have applied the 'Traditional Approach' depicted in Figure 7.1 to the popular P2P protocol, BitTorrent. Considering BT as a strategy in iterated games, and using the concept of 'opportunity costs', we were able to unearth new protocol aspects. Using a different abstraction as compared to previous work [97] under which BT is a Nash equilibrium, we demonstrated that in our abstraction, BitTorrent is not Nash equilibrium. We also devised a protocol variant that is a Nash equilibrium under our assumptions. The Nash equilibrium analysis of BT helped us establish the fact that for game-theoretic analysis of complex protocols, different abstractions will yield different results.

We now need to consider what we gain from our equilibrium analysis and what exactly is meant when a protocol is said to be robust. It was only subsequent to the proof that BitTorrent is a Nash equilibrium [97], that questions about BitTorrent's robustness were raised. Locher *et al.* [73] showed that BitTorrent's TFT policy is vulnerable to attack from an *Always Defect* strategy. Later on, yet another BT exploit was devised based on an adaptive policy for number of partners and variable rate of reciprocation [94]. Even in our case, simply by choosing an abstraction that incorporates the interactions between various classes of peers in more detail, we showed that BT is not a Nash equilibrium.

If an analysis were to be done that includes more details, would our Birds analysis still hold? Can a method be developed that can analyze protocols more comprehensively? Specifically, we would like to know in detail how robust our own Nash variant, Birds, really is. Generally, we aim to understand how to design a protocol that can be quantified as robust. What are its potential adversaries and to what degree can it resist them? In the next section, we present Design Space Analysis, an approach that seeks to answer such questions.

7.2 Design Space Analysis

We wish to design distributed protocols that maximize performance of the system under the assumption that protocol variants may enter the system. We present Design Space Analysis (DSA), a simulation based method, which emphasizes the specification and analysis of a design space, rather than proposing a single protocol. First, we list the key elements of DSA. Then we present the *Performance, Robustness, Aggressiveness* (PRA) quantification, a solution concept within DSA.

7.2.1 Key elements of DSA

We consider the elements that are integral to Design Space Analysis.

Flexible behavioral assumptions. In DSA, we relax behavioral assumptions. Protocols may, in the words of Axelrod [7], "simply reflect standard operating procedures, rules of thumb, instincts, habits, or imitation".

Specification of design space. In DSA, keeping in view that complex protocols have many elements that can be gamed by strategic nodes, a design space should encompass relevant details that can affect the incentive structure. Design space specification occurs at two levels: i) *Parameterization*, which involves determining the salient dimensions of the design space, and ii) *Actualization*, which involves specifying a host of actual values for every individual dimension.

The specification of the design space can be inspired by consulting the relevant literature and analyzing existing systems. As an example, the Parameterization phase of the design space for *Gossip Protocols* [63] could result in the following salient dimensions: i) Selection function for choosing partners for exchanging data, ii) Periodicity of data exchange iii) Filtering function for determining data to exchange, iv) Record maintenance policy in local database.

The Actualization of this example design space for gossip protocols could be: For Selection Function, following policies could be used : 1) *Random:* Choose partners randomly; 2) *Best:* Choose partners who have given the best service; 3) *Loyal:* Choose most loyal partners; 4) *Similarity:* Choose partners based on similarity; etc. Similarly, different values could be chosen for each of the other dimensions.

An example of specifying a design space, with both the parameterization and actualization phases, will be described in detail in Sections 7.3.1 and 7.3.2 when we apply DSA to P2P file swarming systems.

Systematic analysis of the design space. In DSA, a desired feature of all solution concepts is a systematic exploration of the design space. This exploration could either follow an exhaustive approach, e.g., a parameter sweep, or a heuristic based approach. By a thorough scan of the space, DSA solution concepts can anticipate strategic variants and

Table 7.2: Existing protocols/designs mapped to our generic P2P protocol design space.

Protocol	P2P Replica Storage [107]	GTG [83]	Maze [78]	Pulse [92]	BarterCast [81]	Private BT Communities
Peer Discovery	Gossip based	orthogonal	Central server	Gossip based	Gossip based	Central server
Stranger Policy	Defect if set of partners full	Unconditional cooperation	Initialized with points	Give positive score	Unconditional cooperation	Initial credit
Selection Function	Closest to own profile	Sort on Forwarding Rank	Ranked on points	Missing list, Forwarding list	Rank/Ban according to reputation	Credits or sharing ratio above certain level
Resource Allocation	Equal	Equal	Differentiated according to rank	Equal	orthogonal	Equal / Differentiated according to credits

predict their effects. Heuristic based approaches can provide partial solutions relatively fast, however, without any guarantees on the level of goodness of the measures.

7.2.2 The PRA quantification

We now present the PRA quantification, a solution concept within DSA. We note that other solution concepts within DSA could also be devised. Using PRA, we can characterize any protocol, from a given design space, over three measures (or dimensions). For a given protocol Π, these three particular measures, are:

- Performance - the overall performance of the system when all peers execute Π (where performance is determined by the application);
- Robustness - the ability of a majority of the population executing Π to outperform a minority executing a protocol other than Π;
- Aggressiveness - the ability of a minority of the population executing Π to outperform a majority executing a protocol other than Π.

We formulate a way to assign values to each of the three measures normalized into the range $[0,1]$. Hence the properties of any given Π can be characterized as a point within a three-dimensional Performance, Robustness, Aggressiveness (PRA) space. The importance of these three measures is evident from the literature. Performance and Robustness have been studied extensively [10, 60, 94]. Aggressiveness has not been explicitly studied but the numerous papers that present protocol variants [57, 67, 68] suggest the need for an aggressiveness measure to determine the effectiveness of a new protocol variant in a population of peers following some other protocol(s).

It is desirable, in open systems in which strategic variants can enter, to design protocols which maximize all three measures. However, it can be conjectured that there will often be a trade-off between them. For example, one may design protocols with high performance but low robustness or conversely high robustness and low performance.

We now define more precisely how we can map a given protocol Π, which can be expressed as a point in the design space, to a point in the PRA space; formally, we define a function $S : D \rightarrow [0,1]^3$, where D is the design space.

We assume that for each peer in a system of peers we can calculate a utility which quantifies individual performance. The measure of performance is application specific, such as download speed in P2P file swarming systems. Given this we define the performance P of protocol Π as the sum of all individual utilities in a population of peers executing Π normalized over the entire protocol design space. Hence, $P = 1$ indicates the best performance obtained from any protocol in the design space.

We define the Robustness R for protocol Π as the proportion of all other protocols from the design space that do not outperform Π in a *tournament*. A tournament consists of multiple *encounters* in which protocol Π plays against all other protocols in turn. An encounter is a mixed population of peers executing one of two protocols. The winning protocol is that which obtains the higher average utility for the peers executing it.

Aggressiveness A for protocol Π is defined in the same way as Robustness, but here Π is in the minority.

7.3 Applying DSA to P2P File Swarming Systems

In this section, we describe our methodology for applying DSA to P2P file swarming systems. First, we Parameterize a generic P2P design space. Next, based on this generic space, we Actualize a specific file swarming design space. Subsequently, we apply the PRA quantification on this space. Finally, we present the results of our analysis.

7.3.1 Parameterization of a Generic P2P Protocol Design Space

We have identified the following salient dimensions applicable to a large variety of P2P systems.

Peer Discovery: In order to perform productive peer interactions, it is necessary to find other partners. For example, when a peer is new in the system, looking for better matching partners or existing partners are unresponsive. The timing and nature of the peer discovery policy are the important aspects of this dimension.

Stranger Policy: When interacting with an unknown peer (stranger), past history cannot be used to inform actions. It is therefore necessary to apply a policy to deal with strangers. The way peers allocate resources to strangers is an important aspect of this dimension.

Selection Function: When a peer requires interaction with others this function determines which of the known peers should be selected. This could include, for example, past behavior (through direct experience or reputation system), service availability and liveness criteria.

Resource Allocation: During peer interactions resources must be allocated to the selected peers (given by the selection function). The way a peer divides its resources among the selected peers, defines the resource allocation policy.

Some existing implemented protocols and proposed designs are listed in Table 7.2. A protocol such as Give-to-Get (GTG) [83] employs unconditional cooperation with strangers as its *stranger policy*. A P2P replica storage design [107] presents a stranger policy which defects on strangers when its set of regular partners is full. For implementing the *selection function*, a deployed P2P reputation system like BarterCast [81] ranks peers in order of reputation score and also proposes to ban peers below a certain reputation score. P2P replica storage selects those peers which are closest to the selecting peer's own storage capacity. This selection policy is identical to the one proposed by us for implementing Birds in BitTorrent-like file-sharing systems. For *resource allocation*, Maze [78] allocates resources proportional to rank. These examples are a few implemented systems and proposed designs; a large variety of P2P systems that rely on eliciting cooperation from participating nodes rely on similar dimensions.

7.3.2 Actualization of a Specific P2P Protocol Design Space

We define some specific actualizations of a BitTorrent-like file swarming system, as described in Section 7.1, based on the general design space of Section 7.3.1. The ideas behind these actualizations have been taken directly from, or inspired by, various works on cooperation done in P2P and also some works done in biology and social sciences in general [7,55,95]. We were motivated to take inspiration from other fields, because eliciting cooperation in decentralized settings is a general problem that has been well-studied.

For the **Stranger Policy**, we define three different actualizations and a value h for the number of strangers to cooperate with:

B1) *Periodic:* Give resources to up to a certain number of strangers periodically.
B2) *When needed:* Only give resources to strangers when set of regular partners is not full. This particular implementation has been inspired by [57].
B3) *Defect:* Always defect on strangers, i.e., give nothing to strangers.

We set h, the number of strangers to cooperate with at any given time, to be in the range $[1,3]$. This gives $3 \times 3 = 9$ different stranger policies. We further add one more stranger policy, where the number of strangers is zero. This gives a total of 10 different stranger policies.

We sub-divide the **Selection Function** into three parts: a candidate list, a ranking function over that candidate list, and finally a value k for the number of peers to select from the ranked candidate list.

For the 'Candidate List' we define two actualizations:

C1) *TFT*, used by default in BitTorrent, using which a peer only places those peers in the candidate list who reciprocated to it in the last round.
C2) *TF2T*, using which a peer places those peers in the candidate list who reciprocated to it in either of the last two rounds. TF2T has been taken from [7].

For the 'Ranking Function', we define six different actualizations:

I1) *Sort Fastest*, ranks peers in order of fastest first.
I2) *Sort Slowest*, ranks peers in order of slowest first.
I3) *Sort Based on Proximity*, ranks peers in order of proximity to one's own upload bandwidth, as in Birds.
I4) *Sort Adaptive*, ranks peers in order of proximity to an *aspiration level*, which is adaptive and changes based on a peer's evaluation of its performance. This has been inspired from [95].
I5) *Sort Loyal*, ranks peers in order of those who have cooperated with the peer for the longest durations. This has been inspired by [55].
I6) *Random*, does not rank peers and chooses them randomly. This has been inspired by [67].

After applying the ranking function, a peer chooses the k top peers, where k is the maximum number of partners that a peer can have. Currently, we set k to be in the range $[1,9]$. This results in $2 \times 6 \times 9 = 108$ different possibilities for the selection function. We further add one more protocol, where the number of selected peers is zero. This gives a total of 109 different selection policies.

For **Resource Allocation**, we define three actualizations:

R1) *Equal Split*, gives all selected peers equal resources (upload bandwidth).
R2) *Prop Share*, gives others proportional to what they gave in the past. This has been inspired by [68].
R3) *Freeride*, gives nothing to partners.

Based on the above, the total number of unique protocols comes to $10 \times 109 \times 3 = 3270$. We note that this number can be larger or smaller based on what, and how many, specific implementations in the space, designers want to explore[4]. Our purpose here is to show the practicality of the DSA analysis by analyzing a considerable space of unique protocols.

[4]For instance, we do not consider variants for resource allocation to strangers. Also, we do not consider Peer Discovery.

7.3.3 Conducting the PRA quantification

First we describe our simulation model. Then we discuss our methodology for conducting the PRA quantification on the design space described in Section 7.3.2.

Simulation Model

We use a cycle-based simulation model, in which time consists of *rounds*. For *peer discovery* we assume that all peers can connect to each other. In each round, a peer decides to upload to a given number of peers based on some *selection* criterion. It uses its *resource allocation* policy to decide how much to give to each of the selected partners. Furthermore, it decides to cooperate with strangers based on its *stranger policy*. A peer also maintains a short history of actions by others. At the same time a peer also has some rate of requesting services from other peers that depends on specific actualizations. This is the basic model on top of which we explore the design space of Section 7.3.2. We run our simulation experiments with 50 peers, which is a good approximation of an average BitTorrent swarm-size [54]. These peers interact with each other for 500 rounds. We use a cluster for running our experiments. In order to lend realism to our experiments, we initialize the peers using the bandwidth distribution provided by Piatek *et al.* [94]. We assume that all peers always have data that others are interested in.

Methodology

Based on the PRA quantification, as described in Section 7.2.2, we first measure the Performance of each protocol in the space. For each protocol Π, we run simulations in which all peers execute Π and measure the average performance of the population. We perform 100 runs for each protocol. In these experiments we define average performance as throughput of the population.

Next we run Robustness experiments. We run simulations, where each protocol plays against every other protocol. We refer to a competition in which two protocols are pitted against each other as an *encounter*. For each encounter, the peer population is split up into two equal halves where half the peers execute Π and the other half executes another protocol. We chose 50% because this is the highest number that an invading protocol can have. Anything higher than 50% means that the invading protocol actually becomes the majority protocol. We hypothesize that if a protocol is robust when 50% of the population executes another protocol, then it will be robust against small invading populations. To verify this hypothesis, we also conduct simulations with the population split up into 90-10, where 90% of the peers follow protocol Π, while 10% execute other protocols in the space, and observe similar results (the Pearson's correlation coefficient of the two sets of results is 0.97). We do 10 runs for each particular encounter between two protocols.

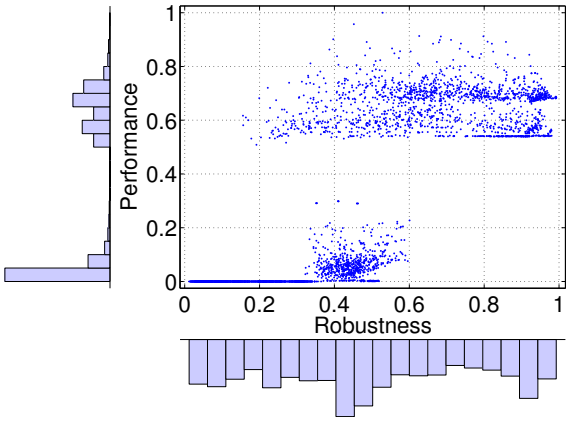


Figure 7.4: Scatter plot of all 3270 protocols in the design space with Robustness against Performance. The results presented here are a synthesis of over 107 million individual simulation runs. Histograms are also shown.

This means that a protocol Π plays against the same protocol ten times. For each run, we compare the average performance of Π with the average performance of the other protocol. If the performance of Π is greater than the performance of the other protocol, we mark it as a *Win* for Π, otherwise we mark it as a *Loss* for Π. The robustness value for Π is calculated by number of games that it wins against all opponents in all runs divided by the total number of games that it plays, which is constant for all protocols.

For Aggressiveness we use the same setup as for Robustness experiments, with the difference that the population is so divided that 10% of the peers execute Π while the rest execute another protocol.

Next, we present results of applying the PRA quantification over the design space[5] described in Section 7.3.2.

7.3.4 Results and Discussion

Figure 7.4 shows all the 3270 protocols actualized in Section 7.3.2, with their normalized Robustness and Performance values. Given the methodology of conducting the PRA quantification as described in Section 7.3.3, this figure represents a synthesis of 107 million individual runs. For performance, each point represents the average normalized performance of a protocol Π, over 100 runs. The robustness values are calculated as described in Section 7.3.3. The maximum variance in the runs for each protocol's performance and robustness was as low as 0.0014 and 0.0206, respectively.

Performance. We shall start by looking at different regions of Figure 7.4. It can be seen

[5]We note here that the entire series of simulations for PRA took around 25 hours on a 50-node dual-core cluster.

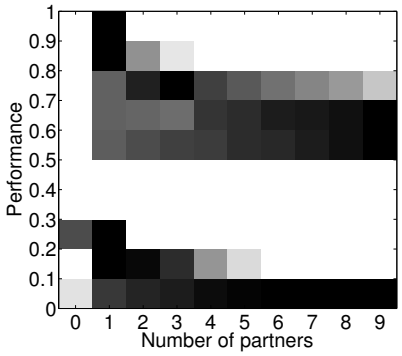


Figure 7.5: Normalized Performance histograms for different partner values. Darker squares represent high 'partner value' frequency for a particular Performance interval.

that numerous protocols have both very low performance and robustness in the range $[0, 0.2]$ (as can be seen from the large histogram bars in the bottom left hand corner). Upon inspection we discover that most of them are freeriders, although different kinds of freeriders. The freeriders with low robustness usually defect on strangers, while freeriders with low performance usually defect on their partners. The maximum performance that such freeriders get is 0.31. This can be seen in the form of two clusters in Figure 7.4, where the lower and upper clusters are below and above 0.4 performance, respectively.

The lower cluster for performance contains all freeriders that do not reciprocate to their partners. It also contains some freeriders that defect on strangers. The upper cluster for performance does not contain any freeriders that defect on their partners but interestingly does contain freeriders who defect on strangers. In fact, the protocol with the highest performance (of 1), is the one that always defects on strangers, has the *Sort Slowest* ranking function, and maintains one partner. We try to dissect why this particular protocol performs so highly. By defecting on a stranger, a protocol *in effect* uploads nothing to the stranger. Since, all peers follow the *Sort Slowest* ranking function, a peer p_1 on downloading nothing from another peer p_2, immediately discards its partner p_3 and chooses p_2 to be its partner. Peer p_3 now finds itself partner-less. However, it can also quickly find a partner for itself by uploading nothing to another peer. Thus by following this protocol, peers rarely find themselves without a fully occupied partner set and can always download at full speed.

Thus, counter-intuitively, by maintaining a single partner, by not uploading anything to strangers and by employing the *Sort Slowest* ranking function, a very high performing protocol can be designed. It is imperative of course in this case that the *resource allocation* method should *not* be *Prop Share*. This is because *Prop Share* will ensure that a peer on getting nothing from another peer, also uploads nothing in response. If that happens, the entire population that follows this protocol will fail to bootstrap.

In Figure 7.5 we observe that in fact many of the top performing protocols are those

with one partner. In Figure 7.5, for each performance interval, darker squares represent higher relative frequency of the number of partners. The empty white spaces in Figure 7.5 reflect the empty spaces in the scatter plot and the histogram of the Performance values in Figure 7.4. All the top 15 performing protocols maintain one partner each and the overall trend in the top performing protocols shows a low number of partners. In the top hundred performing protocols only 11 maintain more than 2 partners. It can be seen from Figure 7.5, starting from the top, till the performance interval $[0.7, 0.8]$, the frequency of low number of partners is relatively higher than the frequency of high number of partners.

We analyze these high performing protocols with a low number of partners more closely now. We have already discussed the features of the highest performing protocol, which uses the *Sort Slowest* ranking function. However, not all high performing protocols with low number of partners have the same features. Many of them employ the *Sort Loyal* ranking function. With the *Sort Loyal* ranking function, it can be conjectured that peers which come to form cooperative relationships early on, stay committed in their relationships. This stability of relationships increases the performance of the system, because at no time do peers find themselves short of partners.

Apart from this, many such protocols often also use the *When needed* stranger policy. This policy also leads to more committed partnerships. With the *When needed* stranger policy, peers only cooperate with strangers when their set of partners is not full. Thus once its partner set is full, a peer will not cooperate with strangers. By not cooperating regularly with strangers, a peer protects itself from breaking relationships by avoiding temptation. This is because when a peer p defects against a stranger, it does not get back anything in response from the stranger. In this way peer p does not discover how much better or worse the stranger is than its current partners, and thus continues to stick with them.

It could be assumed that in the presence of churn, protocols with low number of partners will not perform so well. However, we ran Performance tests for the whole space under churn rates of 0.01 and 0.1 per round, and found that it was still the protocols that employed a low number of partners that performed the best. Thus, it can be claimed that having low number of partners is a desirable feature of high performing protocols, given that everyone in the population runs the same protocol.

Robustness. It is interesting to compare Figure 7.5 with Figure 7.6. While a low number of partners seems to be a good choice for high performance, the situation is reversed when it comes to robustness. In Figure 7.6, we observe that most of the highly robust protocols have a high number of partners. This can be seen in the top right hand corner of the figure. This is intuitive, because protocols that employ a low number of partners would very likely perform worse in face of an invading protocol that employs high number of partners. This is because, in case of an invasion, peers employing a protocol that maintains a high number of partners are less likely to find themselves short of partners as compared

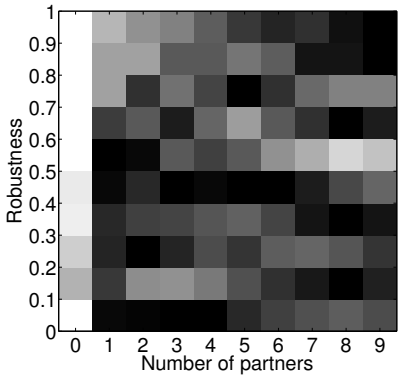


Figure 7.6: Normalized Robustness histograms for different partner values. Darker squares represent high 'partner value' frequency for a particular Robustness interval.

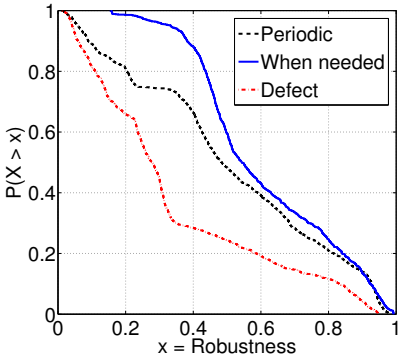


Figure 7.7: Complementary CDF plots of Robustness of different stranger policies.

to peers who follow a protocol that maintains a low number of partners. Hence, in the case that some partners defect, peers with a high number of partners can continue to download at high speeds while peers with a low number of partners are likely to suffer with poor speeds. To put it straightforwardly, a peer that maintains 9 partners will suffer less when one of its partners defects on it, as compared to a peer that only maintains one partner.

As we are considering the robustness of protocols, it can be seen from Figure 7.4 that there are some protocols that are highly robust with robustness values above 0.99. By analyzing these highly robust protocols, we discovered their interesting properties. Including the most robust protocol in that cluster, these protocols use a combination of the *When needed* stranger policy, the *Sort Fastest* ranking function and the *Prop Share* reciprocation function. Figure 7.7 shows that only those protocols which use the *When needed* stranger policy reach robustness levels greater than 0.99. Similarly, it can be seen from Figure 7.9 that protocols which use the *Sort Fastest* ranking function, are the most robust protocols. Figure 7.8, shows that even though the *Equal Split* resource allocation policy does quite well, it is the *Prop Share* policy that manages to get robustness values greater than 0.99.

Since this is a vital point, as it tells us about the features of protocols that are almost completely robust, we consider these properties in detail. As discussed previously, the *When needed* stranger policy only cooperates with strangers when its set of partners is not full. The *Sort Fastest* ranking function, as the name implies, ranks peers in decreasing order of their speed, and finally the *Prop Share* resource allocation policy, allocates resources to peers in proportion to their contribution.

This combination, first of all, validates the claims about the robustness of the *Prop Share* mechanism [68]. However, it should be noted that, unlike their proposal, these protocols do not reciprocate to everyone. In fact, the most robust protocol maintains only 7 partners. Secondly, the combination of *When needed* with *Prop Share* is very important to note. In [68] a cryptographic bootstrapping technique was proposed to avoid exploitation by freeriding strangers. Our results suggest that the combination of the *When needed* stranger policy with the *Prop Share* resource allocation policy is a practical and lightweight alternative for designing robust protocols.

Trade-off between Performance and Robustness. Looking at the very robust protocols, we note that most of them are not among the high performing protocols. This suggests a trade-off between performance and robustness.

However, looking at the top right hand corner of Figure 7.4, we can see that there are at least some protocols that are robust and also have high performance (with robustness and performance values above 0.8). On inspection we find that there are 9 such protocols and *all* of these protocols follow the *Sort Loyal* ranking function. No other dimension (such as resource allocation, stranger policy, etc.) is uniform across all 9 protocols. *Sort Loyal* cooperates with those other peers who cooperate with it for the longest durations. It could have been assumed that a ranking policy like *Sort Loyal* would not fare very high in terms of robustness. This is because of the danger that a fast peer that employs the *Sort Loyal* ranking function, could get stuck with very slow peers (who follow another protocol) that keep cooperating with it. However, it is interesting to note that the highest robustness achieved by a protocol that sorts others based on loyalty, is actually a very high 0.97.

Aggressiveness. In Figure 7.10 we see that Robustness and Aggressiveness are linearly correlated with a Pearson's correlation coefficient of 0.96. This suggests that robust protocols are also very aggressive and there does not seem to be a major payoff between robustness and aggressiveness. We can conclude that the results for Robustness also hold for Aggressiveness.

Regression Analysis

We applied multiple linear regression analysis for the whole protocol design space, which is reported in Table 7.3. Values of h and k (i.e. number of strangers and partners) were

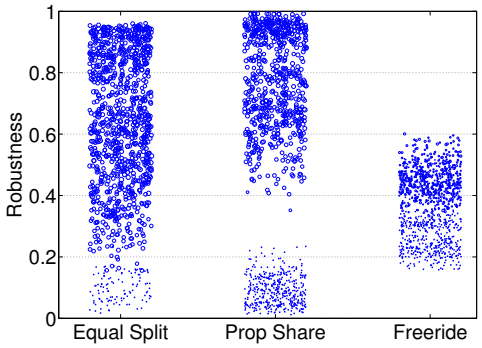


Figure 7.8: Robustness values using different resource allocation methods. Each circle is a unique protocol. Bigger circles represent better performance.

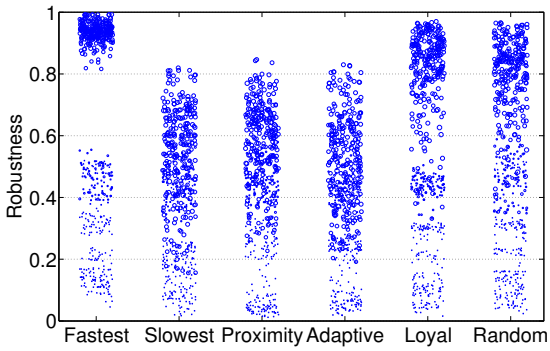


Figure 7.9: Robustness values using different ranking functions. Each circle is a unique protocol. Bigger circles represent better performance.

treated as numerical values (in the table $\tilde{h}$ and $\tilde{k}$ are the standardized values of h and k, respectively), whereas the other variables are categorical, thus were substituted by dummy variables.

These results serve as a summary of our previous results and also examine some dimensions which were not covered in the previous section. From Table 7.3 we can conclude the following: i) Choosing *Freeride* as a resource allocation policy (R3) has the biggest negative impact on Performance and Aggressiveness and it is also has a big negative impact on Robustness. ii) Another bad choice is the *Defect* stranger policy (B3). This policy has the biggest negative effect on Robustness. On the other hand, the *When needed* policy (B2) increases Robustness and Aggressiveness, but is not statistically significant for Performance. iii) Increasing the number of strangers to cooperate with increases all three, Performance, Robustness, and Aggressiveness values and this variable has the biggest positive effect on all three measures. iv) Higher number of partners results in an increase in Robustness and Aggressiveness, but not for Performance. v) We can see that *TF2T* strategy (C2) plays a consistent negative role for all three measures. vi) The choice of the ranking function has a big effect on Robustness and Aggressiveness. This

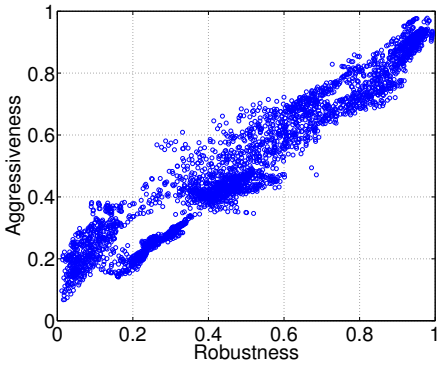


Figure 7.10: Scatter plot of robustness and aggressiveness values of the protocols. The Pearson's correlation coefficient is 0.96.

Table 7.3: Multiple linear regression analysis applied for the PRA measures of the whole search space. The adjusted R^2 values are reported in the second row. The standard errors for all the variables are less than 0.012. Significance level is indicated as 'OK' if it was less than 0.001.

	Performance (adj.$R^2 = 0.68$)			Robustness (adj.$R^2 = 0.52$)			Aggressiveness (adj.$R^2 = 0.61$)		
variable	estimate	t value	sign.	estimate	t value	sign.	estimate	t value	sign.
(intercept)	0.661	64.372	OK	0.813	73.286	OK	0.801	96.300	OK
$\log(\tilde{k})$	−0.008	−2.487	–	0.035	10.334	OK	0.037	14.591	OK
$\log(\tilde{h})$	0.109	33.121	OK	0.115	32.287	OK	0.092	34.340	OK
B2	0.008	1.046	–	0.026	3.010	OK	0.026	4.012	OK
B3	−0.206	−26.257	OK	−0.241	−28.399	OK	−0.190	−29.778	OK
C2	−0.066	−10.567	OK	−0.045	−6.576	OK	−0.039	−7.716	OK
I2	0.023	2.151	–	−0.214	−18.186	OK	−0.212	−24.000	OK
I3	0.020	1.831	–	−0.199	−16.911	OK	−0.155	−17.503	OK
I4	0.022	1.975	–	−0.230	−19.517	OK	−0.193	−21.796	OK
I5	0.031	2.843	OK	−0.074	−6.262	OK	−0.097	−11.004	OK
I6	−0.009	−0.796	–	−0.082	−7.080	OK	−0.090	−10.259	OK
R2	−0.194	−25.260	OK	−0.093	−11.188	OK	−0.053	−8.511	OK
R3	−0.544	−70.848	OK	−0.220	−26.562	OK	−0.253	−40.697	OK

is in line with Figure 7.9. However, choice of ranking function does not have significant impact on Performance, except for the *Sort Loyal* ranking function (I5), which increases Performance.

Birds according to Design Space Analysis

We devised a robust variant of BitTorrent in Section 7.1.3 using a game-theoretic analysis. Subsequently, we augmented game-theoretic analysis for protocol design with Design Space Analysis. We would now like to inspect if the results that we obtained from our game-theoretic analysis have held. How does Birds fare in the larger design space, under a more comprehensive and more realistic analysis?

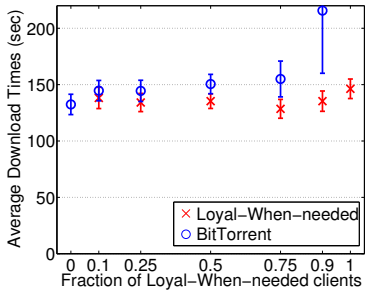
For the Performance measure, the best Birds variant, i.e., a protocol that at the very least ranks other by *Proximity* and employs *Equal Split* reciprocation, does well with a Performance value of 0.83 to rank at 30 among all 3270 protocols. In Robustness, Birds achieves a highest value of 0.76 and ranks at 714. For Aggressiveness, Birds achieves a highest value of 0.74 to rank at 630 among all protocols.

Discussion

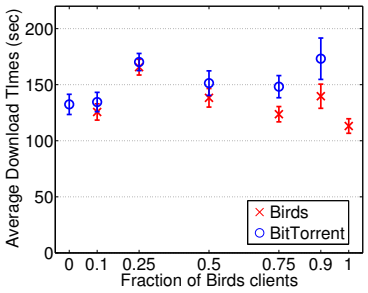
Using DSA we were able to discover protocol variants that do well by employing interesting and counter-intuitive combinations of actualized dimensions. Some combinations lead to extremely high performance, while some lead to very high robustness. We also discovered a few protocols that are both highly robust and also have high performance.

The highly robust protocols are the best candidates for usage in open distributed systems, in which protocol variants may enter; whereas the protocols that achieve very high performance are perhaps more suited to closed systems, where incentives are not required. With the regression analysis on the whole design space we were able to measure the relative impacts of the different dimensions. This also gives new insights into practical protocol designs, and indicates the actualized dimensions that should be preferred and those that should be avoided.

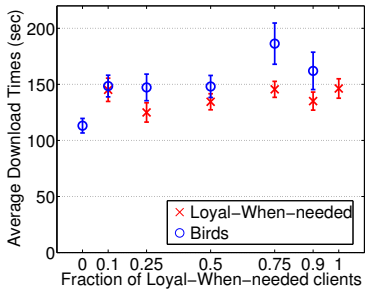
Finally, we observed that Birds ranked very well in Performance and it is within the top quarter in Robustness. Given the fact that Birds was devised using a highly abstracted game-theoretic analysis, we claim to have: a) shown that a game-theoretic analysis is a useful tool which can be used to devise protocols through simple abstractions, that do reasonably well; and b) demonstrated the utility of DSA, by discovering *several* protocols that do better than Birds, in terms of Robustness and Performance.



(a)



(b)



(c)

Figure 7.11: Encounters between three selected protocols.

7.4 Validation of DSA Results

We discovered several interesting protocols using Design Space Analysis. In this section we want to explore how well DSA can be used to design deployable protocols. In order to prove the feasibility of using DSA to produce robust and high performing protocols that can be deployed, we modified an instrumented BitTorrent client provided by [65]. From the discovered protocols we choose one that uses the *Sort Loyal* ranking function and the *When needed* stranger policy because this variant resulted in both high Performance and Robustness according to DSA. We term this protocol as 'Loyal-When-needed'. Another variant was suggested by our Nash equilibrium analysis, which is the Birds protocol. Birds uses the *Proximity* ranking function. The third protocol type is the standard BitTorrent, which represents the baseline.

Experimental setup. In our experiments we pitted one protocol against another, with varying proportions of the two protocols. We adopted an experimental setup similar to [68] and [94]. The total number of leechers is 50. We setup one seeder with an upload bandwidth of 128 KBps. We setup a local tracker, with peers downloading 5MB files. We used the bandwidth distribution provided by [94]. Peers leave upon completing their download. The results of these runs can be seen in Figures 7.11 and 7.12, where each point in the figures represents the average over at least *10* runs and error bars mark 95% confidence intervals. Finally, we base our decision to use a cluster on the arguments and

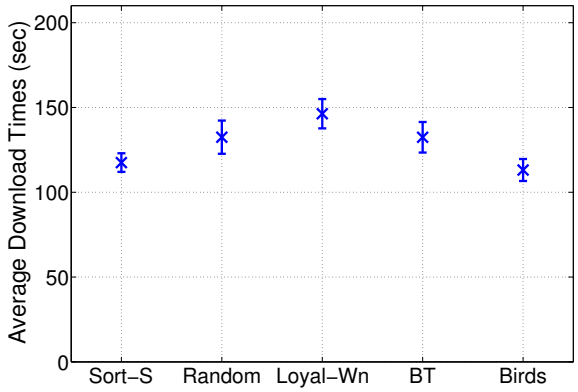


Figure 7.12: Performance of various protocols.

results in [103].

BitTorrent vs Loyal-When-needed. Figure 7.11(a) represents competitive encounters between BitTorrent and Loyal-When-needed clients. We can see the consistent trend of the Loyal-When-needed clients regarding the average download times: it is largely independent of the composition of the swarm. Moreover, Loyal-When-needed clients never do worse than BitTorrent, and they do significantly better if they are in a majority in the swarm.

Birds vs BitTorrent. Figure 7.11(b) represents a validation of our game-theoretic analysis from Section 7.1.3: Birds does as well or better than BitTorrent in all proportions. The difference is statistically significant if the proportion of Birds clients is more than or equal to three quarters. We can also conclude that the swarm with only Birds clients results in significantly better average download time than a swarm with only BitTorrent clients.

Birds vs Loyal-When-needed. Finally, in Figure 7.11(c) we compare the competitive encounters of Birds and Loyal-When-needed. In line with our DSA results, the Loyal-When-needed protocol is more robust than Birds. The degradation in Birds performance becomes statistically significant when the two protocols compete; this is more evident when the Loyal-When-needed clients are in the majority.

Performance of various protocols. We now compare the performance of different protocols when all peers in the population execute the same protocol. For this, we consider two additional protocols that we discovered through DSA. Figure 7.12 shows that a protocol that we term Sort-S, together with Birds, fares the best when all peers in the swarm follow the same protocol. Sort-S is the interesting protocol that we discovered in our DSA analysis. It uses the *Sort Slowest* ranking function, defects on strangers and only has one partner. It is interesting to observe that a protocol that uses the *Sort Random* ranking function performs as well as BitTorrent. This recalls the results presented in [67]. We note that Figure 7.12 does not say anything about the Robustness of these protocols.

7.5 Related work

The study of incentive mechanisms for distributed systems has been a well-studied topic in the research community. Such works can be roughly categorized into the works, which came before the seminal papers by Feigenbaum and Shenker [27], and Dash *et al.* [21], and those that came subsequently. We concentrate on the latter works. Mahajan *et al.* [75] described their unsuccessful efforts at applying game theory for system design. They suggested that relaxing the notion of perfect selfishness could increase the applicability of game theory. We demonstrated that game theoretic analysis despite high level of abstraction, can result in a fruitful analysis, which, however, needs to be augmented with a detailed analysis.

Feldman *et al.* [32] applied an evolutionary game-theoretic analysis on a P2P design space. Their analysis differed from ours as it focussed on simple cooperation, defection and reciprocation, whereas we analyze a considerably vast space of protocols. Also, we apply the results of our analysis to BitTorrent to validate our approach.

Qiu *et al.* [97] under certain assumptions showed that BitTorrent is a Nash Equilibrium. We used a different abstraction and proved otherwise. Similarly Levin *et al.* [68] argued that BitTorrent is an auction, and with their model gained insights to develop a more robust variant. We showed that considering BitTorrent as a strategy in a game, also leads to insights, and we also developed a more robust variant.

Finally, many papers propose protocol variants not based on game-theoretic analysis [10, 67], but to our knowledge, ours is the first attempt to provide a comprehensive, alternate approach that can complement game-theoretic analysis.

7.6 Conclusion

We have presented Design Space Analysis (DSA), a simulation based approach that complements game-theoretic analysis of incentives in distributed protocols. DSA emphasizes the specification and analysis of a design space, rather than proposing a single protocol. We have shown that DSA can be used to gain an in-depth analysis of the properties of protocol variants, and it can be used for designing deployable protocols. For future work, we would like to explore if a solution concept similar to PRA quantification could be developed which explores the design space using a heuristic based approach. This could be needed in situations where a thorough scan of the design space becomes infeasible due to its size. We would also like to test DSA on distributed domains other than P2P. In conclusion, we note that DSA is a general method, which is practical and easy to apply, and we have demonstrated its merits by applying it successfully to P2P file swarming systems.

Chapter 8

Conclusion

In this thesis, we explored the various ways in which socioeconomic ideas have been applied in P2P file sharing systems. In doing so, we made contributions in three different ways. a) We identified new problem areas that could benefit from the application of socioeconomic ideas; b) We identifed problems in the existing usage of socioeconomic ideas in the literature; and c) We introduced novel approaches that can alleviate the socioeconomic problems in P2P systems.

In this chapter, we present our conclusions, which serve as the answers to the research questions identified in the introduction, and also propose lines of work that can be explored in the future.

8.1 Conclusions

We list the salient conclusions that we derived from our works.

1) Using an intuitive voting mechanism that is based on an 'experience function', we showed that peers are able to locate and rank high quality content in a decentralized manner. The inspiration behind the 'experience function' is that normally people who contribute more to the community are more visible and can make their 'voice heard' more in comparison to those who are less active. Also, the idea is that the opinions of people who have invested relatively more in the community, should be given more weight than those of newly arriving peers. In this way, this mechanism saves the system from a flash crowd of newly arriving peers who spread malicious votes.

2) Private BitTorrent communities use either credit based or sharing ratio based schemes to provide incentives to peers to seed. Based on anecdotal evidence, we realized that it can be very hard for peers to earn credits or sharing ratios in such communities. Using a simple model, we were able to determine that indeed, such communities can suffer from problems where peers are not able to earn credits or sharing ratios, thus causing the performance of the system to go down. Ironically, we determined that under the presence

of peers with heterogeneous bandwidth capacities, the situation gets even worse. Finally, we showed injecting credits into the system, helps peers seed for shorter durations and consequently improves the system performance.

3) Usually, incentive schemes are developed with one user model in mind. We have analyzed what happens when peers in the system instead of being rational, contribute more to the community than is envisaged by the system designers. We showed that in the presence of "hoarders", peers who contribute more than is required, the system performance in private BitTorrent communities degrades. In order to improve the system performance in such situations, we propose new strategies that effectively impose tax on peers based on the seeder and leecher ratio in the system.

4) It proved to be a fruitful line of investigation to apply effort-based incentives in P2P systems. Our results suggest that effort based incentives can be usefully applied for improving both system efficiency and fairness. The encouraging thing is that an effort based policy proved to be successful in two different domains: the BitTorrent protocol and a model of a private file sharing community that uses a credit based sharing ratio scheme

5) We presented a fleshed out alternative to game-theoretic analysis for modeling incentives in distributed systems. We applied our approach, called Design Space Analysis (DSA) to P2P file swarming systems. We found several interesting protocols using our approach and demonstrated that DSA can be used to build deployable systems. In our view, given our approach, designers should now think beyond game-theory when modeling and analyzing incentives. We believe, we have clearly demonstrated both the strengths and weaknesses of a game-theoretic approach and have shown the way forward for the employment of alternate approaches.

8.2 Future Work

We list the following areas of future work:

1) In devising a voting mechanism that can prevent a flash crowd of malicious attackers, we have used a fixed experience threshold. If attackers decide to bear this cost, then the mechanism would prove to be ineffective. We have outlined a potential approach to address this particular concern, which is using an *adaptive threshold* for the *experience fucntion*. Such an approach could be said to be inspired by the human immune system: The detection of a pathological attack causes the body to protect itself from outside attackers, and revert back to normal when the attack has been vanquished. It is also analogous to human behavior in social systems, where people and institutions with higher reputation, those who are deemed more experienced, are respected and trusted more by the society at large, in times of crisis.

2) It would be worthwhile to study what effects sharing ratio and credit based enforcement have on individual peers performance, such as upload capacity utilization and seeding times. At the same time, a systematic analysis of various strategies used by different community administrators designed to make rare content more 'appealing' to seeders and to help those stuck in unpopular swarms, is an area that should be studied more extensively.

3) A potentially fruitful area of research is to design user models by studying user behavior in P2P file sharing communities. User models are algorithms that attempt to mimic the behavior of users including how they adapt their behaviour over time. Given sufficiently realistic user models it should be possible to evaluate and improve cooperation mechanisms prior to deployment. Potential candidates include: evolutionary learning, satisficing, altruistic punishment, and preferential attachment.

4) Solution concepts which can heuristically scan huge design spaces should be explored that can facilitate the extensive use of 'Design Space Analysis' (or similar approaches) by system designers for the modeling of incentives. An intuitive solution concept that presents itself readily is based on the idea of a knockout tournament. Instead of each protocol playing against the other, a protocol plays against some randomly chosen protocol. The winner of their 'match' proceeds to the next round, and so on. In this way, such a knockout tournament would proceed much like the FIFA world cup or other such tournaments.

Appendix A

Proof of BitTorrent not being an NE

In the following we use the notation introduced in Table 7.1 and the results from Sections 7.1.2 and 7.1.3.

In order to show that *BitTorrent is not a Nash equilibrium* (NE), we consider a swarm with $N-1$ BitTorrent (BT) peers and assume that one peer using the Birds protocol enters this swarm.

In this setup the expected number of games won by the BT clients against higher and lower classes do not change. On the other hand, for the Birds client only the formula for the expected number of games won against the peers from the lower classes changes to $E_B^r[B \to c]' = N_B/N_r$, which is the same as for the BT clients.

Now, we consider the class C where the peer using the Birds protocol is located. The expected values of the number of games that the peers win due to reciprocation from other peers in this class will be $E_B^r[C-c]' = U_r - K$ for Birds and

$$\begin{aligned} E^r[C \to c]' &= \frac{N_{C'} - U_r}{N_{C'}}(U_r - K - E[A \to c]) \\ &\quad + \frac{U_r}{N_{C'}}(U_r - E[A \to c] - K') \\ &= U_r - K - E[A \to c] - \frac{U_r}{N_{C'}}(K + K') \end{aligned}$$

for the BT clients, where $N_{C'} = N_C - 1$, and $K' = 1 - \left((1 - E[A \to c])(1 - \frac{1}{U_r})\right)^{U_r - 1}$, which leads us to the fact that

$$E_B[C \to c]' > E[C \to c]'.$$

Regarding the 'free game wins', the formulae change to

$$E_B[C \to c]' = \frac{N_{C'}}{N_C}(N_C - E^r[C \to c]')/N_r,$$

$$E[C \to c]' = E_B[C \to c]' + \frac{N_C - E_B^r[C \to c]'}{N_C N_r},$$

which says that $E[C \to c]' > E_B[C \to c]'$; however,

$$E_B^r[C \to c]' + E_B^r[C \to c]' > E^r[C \to c]' + E[C \to c]'$$

holds. Thus, the peer using the Birds protocol, on average, wins more games than any of the BT clients, proving that BT is not a NE.

Proof of Birds being an NE

Now we show that *it is a NE when all peers in the swarm follow the Birds protocol.*

We assume that there are $N-1$ peers following the Birds protocol and one peer using the BT protocol enters this swarm. We give a formal proof for the case when this new peer uses BT; the other three cases (regarding class-based reciprocation) can be proved in the similar way.

First, we consider the games where peers get reciprocation. Neither the Birds peers nor the BT peer get anything from the higher and lower classes. For that particular class C, where the BT peer is located we have

$$\begin{aligned} E_B^r[C \to c]'' &= \frac{N_{C'} - U_r}{N_{C'}} U_r + \frac{U_r}{N_{C'}}(U_r - E[A \to c]) \\ &= U_r - \frac{U_r}{N_{C'}} E[A \to c], \end{aligned}$$

where $N_{C'}$ is the number of Birds in class C, i.e. $N_{C'} = N_C - 1$. Moreover, we have $E^r[C \to c]'' = U_r - E[A \to c]$; from here it is easy to see that $E_B^r[C \to c]'' > E^r[C \to c]''$.

'Free game wins' remain the same. The expressions for the same class become

$$E[C \to c]'' = \frac{N_{C'}}{N_C} \times \frac{N_{C'} - E_B^r[C \to c]}{N - U_r - 1},$$

and

$$E_B[C \to c]'' = E[C \to c]'' + \frac{N_{C'} - E^r[C \to c]}{N_{C'}(N - U_r - 1)},$$

Thus we conclude that $E_B[C \to c]'' > E[C \to c]''$ which completes our proof that Birds is a NE.

Appendix B

On why BitTorrent does and does not employ TFT; and why reading literature from other fields is essential for P2P designers working on incentives.

It has been suggested BitTorrent is an auction [68]. In BitTorrent peers only reciprocate with some x others who offer them the highest bandwidth. And in this sense BitTorrent can be claimed to be an auction. In BitTorrent, the fastest first strategy, in non-converged state, leads to a peer defecting against slower peers as soon as it finds better partners. And in converged state (the state when a peer has found adequate number of equal partners) a peer *always* defects against slow peers and only deals with peers of one's own bandwidth class and higher. This fact that peers can choose partners to reciprocate with, together in particular with the fact that a peer in BitTorrent does not reciprocate with all partners that cooperate with it, has led to questions about BitTorrent's claim to be using TFT [68]. However, while in the P2P community there have been justified questions about BitTorrent unorthodox adaptation of TFT, such adaptations have been frequently performed in various other fields as well. So for example, a body of work abounds on partner choice models in IPD [8, 19, 24, 34, 119]. Similarly, such games in which peers have unequal resources, have been studied in the Social Sciences literature, for example in [34, 45]. These studies show that under the condition where players are unequal, class segregation can be observed [45]. Such works then anticipate the clustering in BitTorrent phenomenon observed in [65].

Strategies such as Pavlov (Win-stay, lose-shift) and Out-for-Tat (OFT) are also reminiscent of BitTorrent. In such strategies, peers exit the game with a partner who has defected and search for better partners. Jin, Hayashi, and Shinotsuka [13,44,89], report results of a laboratory IPD experiment with and without selective play. In the selective- play condition, subjects in four-person groups could express their rank-ordered preferences for partners. Jin et al. observed widespread use of an OFT strategy. Eventually, players who preferred one another formed committed relationships marked by much higher levels of cooperation than observed without selective play. This is similar to the convergence and clustering among classes of different peers in BitTorrent as demonstrated by [17, 65, 82].

Writing in the Journal of Theoretical Biology, Hruschka et al. [55] propose a model of IPD, where player's memories are restricted to one number of K different partners. This recalls the number of regular slots in BitTorrent, which are awarded to peers with the highest upload rates averaged over the last x rounds.

So, the fact that BitTorrent uses tit-for-tat but not exactly in the manner (or setting) as it was elucidated in [7], does not necessarily mean BitTorrent is not tit-for-tat and one can find counterparts in several studies in evolutionary biology, psychology and sociology that have employed the tit-for-tat strategy with new flavors and settings.

So for example the effects of tit-for-tat have been studied where: there are unequal partners [34, 45]; where players have a choice to refuse playing with other players [119]; and finally where based on their preference, peers can keep up to K different partners for regular interaction [55] (this is akin to the number of regular unchoke slots in BitTorrent).

Our point is that there is a wealth of literature on IPD type games with various strategies, some of which very similar to BitTorrent. Also, such studies have anticipated effects such as class segregation, discovered later in BitTorrent. Even a cheating client such as BitThief can easily be explained as a "defect only" strategy that exploits *nice* strategies such as TFT, Pavlov and Out-for-tat.

Summing up, there is a vast amount of literature on games such as IPD with players using strategies ranging from *tit-for-tat* to *out-for-tat*, etc. Such works have already shown results discovered later in the P2P literature, such as clustering in BitTorrent [65] among other things. Therefore, in Chapter 7, we presented an analytical model of BitTorrent as a TFT like strategy in a series of IPD games. Unlike previous work which relates the Prisoner's Dilemma to BitTorrent, we recognize the heterogeneity of peers in real communities and model the peers' payoffs accordingly. More importantly, we realize that unlike traditional strategies, BitTorrent "unlocks the doors of the Prison" [13], where peers are free to choose other partners upon facing defection. Also, as mentioned earlier, for the actualization of the design space in Section 7.3.2, we took inspiration from works done in fields such as biology, psychology, and social sciences in general.

Bibliography

[1] E. Adar and B. A. Huberman. Free Riding on Gnutella. Technical report, Xerox PARC, August 2000.

[2] M. Albert. *Parecon: Life after capitalism*. Verso Books, 2003.

[3] N. Andrade, F. Brasileiro, W. Cirne, and M. Mowbray. Discouraging free riding in a peer-to-peer cpu-sharing grid. 2004.

[4] N. Andrade, E. Santos-Neto, F. Brasileiro, and M. Ripeanu. Resource demand and supply in BitTorrent content-sharing communities. *Computer Networks*, 53(4):515–527, 2009.

[5] P. Antoniadis, C. Courcoubetis, and R. Mason. Comparing economic incentives in peer-to-peer networks* 1. *Computer networks*, 46(1):133–146, 2004.

[6] S. Arteconi. Evolutionary Methods for Self-Organizing Cooperation in Peer-to-Peer Networks. 2008.

[7] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[8] I. Back and A. Flache. The viability of cooperation based on interpersonal commitment. *Journal of Artificial Societies and Social Simulation*, 9:12, 2006.

[9] A. Bellissimo, B.N. Levine, and P. Shenoy. Exploring the use of BitTorrent as the basis for a large trace repository. *University of Massachusetts Technical Report*, pages 04–41, 2004.

[10] A.R. Bharambe, C. Herley, and V.N. Padmanabhan. Analyzing and improving a BitTorrent network's performance mechanisms. In *INFOCOM*, 2006.

[11] A.R. Bharambe, C. Herley, and V.N. Padmanabhan. Analyzing and improving BitTorrent performance. In *Proc. of IEEE Infocom*. Citeseer, 2006.

[12] BitThief. A free riding BitTorrent client. `http://www.bitthief.org`.

[13] R.T. Boone and M.W. Macy. Unlocking the doors of the prisoner's dilemma: Dependence, selectivity, and cooperation. *Social Psychology Quarterly*, 62(1):32–52, 1999.

[14] C. Buragohain, D. Agrawal, and S. Suri. A game theoretic framework for incentives in P2P systems. In *P2P 2003*, pages 48–56, 2003.

[15] L. Buttyán and J.P. Hubaux. Enforcing service availability in mobile ad-hoc WANs. In *Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC. 2000 First Annual Workshop on*, pages 87–96. Ieee, 2000.

[16] X. Chen, Y. Jiang, and X. Chu. Measurements, Analysis and Modeling of Private Trackers. In *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, pages 1–10. IEEE, 2010.

[17] A.L.H. Chow, L. Golubchik, and V. Misra. BitTorrent: An extensible heterogeneous model. In *Proc. of INFOCOM*, 2009.

[18] B. Cohen. Incentives Build Robustness in BitTorrent. In *Proc. of P2PECON*, 2003.

[19] R. C. Connor. Egg-trading in simultaneous hermaphrodites: an alternative to tit-for-tat. *Journal of Evolutionary Biology*, 5:523–528, 1992.

[20] E. Damiani, D.C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 207–216. ACM, 2002.

[21] R.K. Dash, N.R. Jennings, and D.C. Parkes. Computational-mechanism design: A call to arms. *IEEE Intelligent Systems*, 18:40–47, 2003.

[22] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12. ACM, 1987.

[23] J. Douceur. The sybil attack. *Peer-to-peer Systems*, pages 251–260, 2002.

[24] M. Enquist and O. Leimar. The evolution of cooperation in mobile organisms. *Animal Behaviour*, 45:747–757, 1993.

[25] B. Fan, D.M. Chiu, and J.C.S. Lui. The delicate tradeoffs in BitTorrent-like file sharing protocol design. In *Proc. of ICNP*, 2006.

[26] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 1–13. Citeseer, 2002.

[27] J. Feigenbaum and S. Shenker. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. In *ACM DIALM*, 2002.

[28] M. Feldman and J. Chuang. Overcoming free-riding behavior in peer-to-peer systems. *ACM SIGecom Exchanges*, 5(4):41–50, 2005.

[29] M. Feldman, K. Lai, J. Chuang, and I. Stoica. Quantifying disincentives in peer-to-peer networks. In *1st Workshop on Economics of Peer-to-Peer Systems*. Citeseer, 2003.

[30] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 102–111. ACM, 2004.

[31] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for Peer-to-Peer networks. In *Proc. of ACM EC*, May 2004.

[32] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *Conference on Electronic Commerce*, pages 102–111, 2004.

[33] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and whitewashing in peer-to-peer systems. *Selected Areas in Communications, IEEE Journal on*, 24(5):1010–1019, 2006.

[34] A. Flache and R. Hegselmann. Rationality vs. learning in the evolution of solidarity networks: A theoretical comparison. *Computational & Mathematical Organization Theory*, 5:97–127, 1999.

[35] P. Ganesan and M. Seshadri. On cooperative content distribution and the price of barter. 2005.

[36] P. Garbacki, D. H. J. Epema, and M. van Steen. An amortized tit-for-tat protocol for exchanging bandwidth instead of content in P2P networks. In *Proc. of SASO*, July 2007.

[37] Gnutella. `http://rfc-gnutella.sourceforge.net`.

[38] D. Grolimund, L. Meisser, S. Schmid, and R. Wattenhofer. Havelaar: A robust and efficient reputation system for active Peer-to-Peer systems. In *Proc. of NetECON*, June 2006.

[39] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measurements, analysis, and modeling of BitTorrent-like systems. In *Proc. of IMC*, 2005.

[40] R. Hahnel. *The ABCs of political economy*. Pluto Press, 2002.

[41] R. Hahnel and Academic Library. *The ABCs of political economy: A modern approach*. Pluto Press, 2002.

[42] D. Hales and S. Patarin. How to cheat bittorrent and why nobody does. *University of Bologna, Italy, Tech. Rep. UBLCS-2005-12*, 2005.

[43] D. Hales, R. Rahman, B. Zhang, M. Meulpolder, and J. Pouwelse. BitTorrent or BitCrunch: Evidence of a credit squeeze in BitTorrent? In *Enabling Technologies: Infrastructures for Collaborative Enterprises, 2009. WETICE'09. 18th IEEE International Workshops on*, pages 99–104. IEEE, 2009.

[44] N. Hayashi. From Tit-for-Tat to Out-for-Tat. *Sociological Theory and Methods*, 8:19–32, 1993.

[45] R. Hegselmann and A. Flache. Understanding complex social dynamics: A plea for cellular automata based modelling. *Journal of Artificial Societies and Social Simulation*, 1, 1998.

[46] V. Heinink. Metadata Infrastructure for Video on Demand.

[47] Bitcoin homepage. Bitcoin p2p virtual currency. `http://www.bitcoin.org/`.

[48] Bitsoup homepage. Bitsoup the right site for the best torrent byte. `http://bitsoup.org/`.

[49] Dalesa homepage. Dalesa peer-to-peer web cache. `http://www.dalesa.lk/`.

[50] Spotify homepage. A world of music. `http://www.spotify.com/`.

[51] Supernova homepage. Supernova the universal web show source. `http://www.suprnova.org/`.

[52] TV Torrents homepage. Tv torrents collecting the opiate of the masses. `http://tvtorrents.com/`.

[53] Yacy homepage. Yacy the peer-to-peer search engine. `http://yacy.net/`.

[54] Tobias Hoßfeld, Frank Lehrieder, David Hock, Simon Oechsner, Zoran Despotovic, Wolfgang Kellerer, and Maximilian Michel. Characterization of BitTorrent swarms and their distribution in the Internet. *Computer Networks, http://dx.doi.org/10.1016/j.comnet.2010.11.011*, 2010.

[55] D.J. Hruschka and J. Henrich. Friendship, cliquishness, and the emergence of cooperation. *Journal of Theoretical Biology*, 239:1–15, 2006.

[56] M. Izal, G. Urvoy-Keller, E.W. Biersack, P.A. Felber, A. Al Hamra, and L. Garces-Erice. Dissecting BitTorrent: Five months in a torrents lifetime. *Passive and Active Network Measurement*, pages 1–11, 2004.

[57] R. Izhak-Ratzin. Collaboration in BitTorrent systems. In *IFIP-TC Networking*, pages 338–351, 2009.

[58] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems (TOCS)*, 23(3):219–252, 2005.

[59] M. Jelasity, S. Voulgaris, R. Guerraoui, A.M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM Transactions on Computer Systems*, 25(3), 2007.

[60] Seung Jun and Mustaque Ahamad. Incentives in BitTorrent induce free riding. In *P2PECON*, 2005.

[61] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM, 2003.

[62] I.A. Kash, E.J. Friedman, and J.Y. Halpern. Optimizing scrip systems: efficiency, crashes, hoarders, and altruists. In *Proc. of ACM EC*, 2007.

[63] A.-M. Kermarrec and M. van Steen, editors. *ACM SIGOPS Operating Systems Review 41, Special Issue on Gossip-Based Networking*. 2007.

[64] K. Lai, M. Feldman, I. Stoica, and J. Chuang. Incentives for cooperation in peer-to-peer networks. In *Workshop on economics of peer-to-peer systems*. Citeseer, 2003.

[65] A. Legout, N. Liogkas, E. Kohler, and L. Zhang. Clustering and sharing incentives in BitTorrent systems. In *Proc. of SIGMETRICS*, 2007.

[66] A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest first and choke algorithms are enough. In *Proc. of IMC*, 2006.

[67] B. Leong, Y. Wang, S. Wen, C. Carbunaru, Y.M. Teo, C. Chang, and T. Ho. Improving peer-to-peer file distribution: winner doesn't have to take all. In *ACM APSys*, pages 55–60, 2010.

[68] D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee. BitTorrent is an auction: analyzing and improving BitTorrent's incentives. In *Proc. of SIGCOMM*, 2008.

[69] J. Liang, R. Kumar, Y. Xi, and K.W. Ross. Pollution in p2p file sharing systems. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 1174–1185. IEEE, 2005.

[70] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang. Exploring the robustness of BitTorrent peer-to-peer content distribution systems. *Concurrency and Computation: Practice and Experience*, 20(2):179–189, 2008.

[71] Z. Liu, P. Dhungel, D. Wu, C. Zhang, and K.W. Ross. Understanding and improving incentives in private p2p communities. In *Proc. 30th International Conference on Distributed Computing Systems, ICDCS, Genoa, Italy*. Citeseer, 2010.

[72] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free riding in BitTorrent is cheap. In *Proc. of HotNets-V*, 2006.

[73] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free riding in BitTorrent is cheap. In *HotNets-V*, 2006.

[74] R.T.B. Ma, S.C.M. Lee, J.C.S. Lui, and D.K.Y. Yau. An incentive mechanism for P2P networks. In *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pages 516–523. IEEE, 2004.

[75] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Experiences applying game theory to system design. In *ACM PINS*, pages 183–190, 2004.

[76] G.J. Mailath. Do people play Nash equilibrium? Lessons from evolutionary game theory. *Journal of Economic Literature*, 36(3):1347–1374, 1998.

[77] S. Marti and H. Garcia-Molina. Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks*, 50(4):472–484, 2006.

[78] Maze. `http://sarwiki.informatik.hu-berlin.de`.

[79] M. Meulpolder, L. D'Acunto, M. Capotă, M. Wojciechowski, J.A. Pouwelse, D.H.J. Epema, and H.J. Sips. Public and private BitTorrent communities: A measurement study. In *Proc. of IPTPS*, April 2010.

[80] M. Meulpolder, J.A. Pouwelse, D.H.J. Epema, and H.J. Sips. Bartercast: A practical approach to prevent lazy freeriding in P2P networks. In *Proc. of Hot-P2P (in conjunction with IPDPS)*, 2009.

[81] M. Meulpolder, J.A. Pouwelse, D.H.J. Epema, and H.J. Sips. BarterCast: A practical approach to prevent lazy freeriding in P2P networks. In *IEEE IPDPS*, pages 1–8, 2009.

[82] M. Meulpolder, JA Pouwelse, DHJ Epema, and HJ Sips. Modeling and analysis of bandwidth-inhomogeneous swarms in bittorrent. In *IEEE P2P*, pages 232–241, 2009.

[83] J.J.D. Mol, J.A. Pouwelse, M. Meulpolder, D.H.J. Epema, and H.J. Sips. Give-to-get: Free-riding-resilient video-on-demand in p2p systems. In *SPIE/ACM MMCN*, 2008.

[84] A. Nandi, T.W. Ngan, A. Singh, P. Druschel, and D. Wallach. Scrivener: Providing incentives in cooperative content distribution systems. *Middleware 2005*, pages 270–291, 2005.

[85] Napster. `http://www.napster.com`.

[86] T.W.J. Ngan, A. Nandi, A. Singh, D.S. Wallach, and P. Druschel. On designing incentives-compatible peer-to-peer systems. 2004.

[87] T.W.J. Ngan, D. Wallach, and P. Druschel. Enforcing fair sharing of peer-to-peer resources. *Peer-to-Peer Systems II*, pages 149–159, 2003.

[88] S. Nielson, S. Crosby, and D. Wallach. A taxonomy of rational attacks. *Peer-to-Peer Systems IV*, pages 36–46, 2005.

[89] Jin Nobuhito, Nahoko Hayashi, and Hiromi Shinotsuka. An experimental study of prisoner's dilemma network: The formation of commitment among selective dyads. *Japanese Journal of Experimental Social Psychology*, 33:21–30, 1996.

[90] M.J. Osborne and A. Rubinstein. *A course in game theory*. The MIT press, 1994.

[91] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. 1999.

[92] F. Pianese, J. Keller, and E.W. Biersack. PULSE, a flexible P2P live streaming system. In *INFOCOM 2006*, 2007.

[93] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in BitTorrent. In *Proc. of NSDI*, volume 7, 2007.

[94] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in BitTorrent. In *NSDI*, 2007.

[95] M. Posch. Win-Stay, Lose-Shift Strategies for Repeated Games–Memory Length, Aspiration Levels and Noise. *Journal of Theoretical Biology*, 198:183–195, 1999.

[96] J.A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D.H.J. Epema, M. Reinders, M.R. van Steen, and H.J. Sips. Tribler: A social-based Peer-to-Peer system. *Concurrency and Computation: Practice and Experience*, 20(2):127–138, 2008.

[97] D. Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. *ACM SIGCOMM Computer Communication Review*, 34:367–378, 2004.

[98] R. Rahman, D. Hales, M. Meulpolder, V. Heinink, J. Pouwelse, and H. Sips. Robust vote sampling in a p2p media distribution system. 2009.

[99] R. Rahman, D. Hales, T. Vinko, J. Pouwelse, and H. Sips. No more crash or crunch: sustainable credit dynamics in a P2P community. In *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, pages 332–340. IEEE, 2010.

[100] R. Rahman, M. Meulpolder, D. Hales, J. Pouwelse, D. Epema, and H. Sips. Improving efficiency and fairness in P2P systems with effort-based incentives. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5. IEEE, 2010.

[101] R. Rahman, T. Vinko, D. Hales, J. Pouwelse, and H. Sips. Design space analysis for modeling incentives in distributed systems. In *Proc. of SIGCOMM*, 2011.

[102] A. Ramachandran, A.D. Sarma, and N. Feamster. BitStore: An incentive-compatible solution for blocked downloads in BitTorrent. In *2nd Joint Workshop on Economics of NetworkedSystems and Incentive-Based Computing*. Citeseer, 2007.

[103] A. Rao, A. Legout, and W. Dabbous. Can Realistic BitTorrent Experiments Be Performed on Clusters? In *IEEE P2P*, 2010.

[104] A. Rapoport and A.M. Chammah. *Prisoner's dilemma*. Univ of Michigan Pr, 1965.

[105] E. Rasmusen. *Games and information: An introduction to game theory*. Wiley-Blackwell, 2007.

[106] T. Roughgarden and E. Tardos. How bad is selfish routing. *Journal of the ACM*, 49:236–259, 2002.

[107] K. Rzadca, A. Datta, and S. Buchegger. Replica placement in p2p storage: Complexity and game theoretic analyses. In *ICDCS*, pages 599–609, 2010.

[108] J. Sabater and C. Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24(1):33–60, 2005.

[109] S. Saroiu, P.K. Gummadi, S.D. Gribble, et al. A measurement study of peer-to-peer file sharing systems. In *proceedings of Multimedia Computing and Networking*, volume 2002, page 152. Citeseer, 2002.

[110] S. Saroiu, P.K. Gummadi, S.D. Gribble, et al. A measurement study of peer-to-peer file sharing systems. In *proceedings of Multimedia Computing and Networking*, volume 2002, page 152. Citeseer, 2002.

[111] A. Sen. Markets and freedoms: Achievements and limitations of the market mechanism in promoting individual freedoms. *Oxford Economic Papers*, 45(4):519–541, 1993.

[112] S.J. Shenker. Making greed work in networks: A game-theoretic analysis of switch service disciplines. *IEEE/ACM Transactions on Networking*, 3:819–831, 2002.

[113] J. Shneidman and D. Parkes. Rationality and self-interest in peer to peer networks. *Peer-to-Peer Systems II*, pages 139–148, 2003.

[114] J. Shneidman and D.C. Parkes. Specification faithfulness in networks with rational nodes. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 88–97. ACM, 2004.

[115] J. Shneidman, D.C. Parkes, and L. Massoulié. Faithfulness in internet algorithms. In *Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pages 220–227. ACM, 2004.

[116] M. Sirivianos, J.H. Park, R. Chen, and X. Yang. Free-riding in BitTorrent networks with the large view exploit. In *Proc. of IPTPS*, volume 7. Citeseer, 2007.

[117] M. Sirivianos, J.H. Park, X. Yang, and S. Jarecki. Dandelion: Cooperative content distribution with robust incentives. In *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, page 12. USENIX Association, 2007.

[118] R. Snader and N. Borisov. EigenSpeed: Secure peer-to-peer bandwidth evaluation. In *Proceedings of the 8th international conference on Peer-to-peer systems*, pages 9–9. USENIX Association, 2009.

[119] E. Stanley, D. Ashlock, and M. Smucker. Iterated prisoner's dilemma with choice and refusal of partners: Evolutionary results. In *Advances in Artificial Life*, pages 490–502, 1995.

[120] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, 2001.

[121] D. Stutzbach, D. Zappala, and R. Rejaie. The scalability of swarming peer-to-peer content delivery. *NETWORKING 2005*, pages 15–26, 2005.

[122] Q. Sun and H. Garcia-Molina. Slic: A selfish link-based incentive mechanism for unstructured peer-to-peer networks. In *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pages 506–515. IEEE, 2004.

[123] K. Tamilmani, V. Pai, and A. Mohr. SWIFT: A system with incentives for trading. In *P2P Econ*. Citeseer, 2004.

[124] Y. Tian, D. Wu, and K.W. Ng. On distributed rating systems for peer-to-peer networks. *The Computer Journal*, 51(2):162, 2008.

[125] R.L. Trivers. The evolution of reciprocal altruism. *The Quarterly review of biology*, 46(1):35–57, 1971.

[126] D.A. Turner and K.W. Ross. A lightweight currency paradigm for the P2P resource market. *Proc. electronic commerce research*, 2004.

[127] G. Urvoy-Keller and P. Michiardi. Impact of inner parameters and overlay structure on the performance of BitTorrent. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–6. IEEE, 2006.

[128] V. Vishnumurthy, S. Chandrakumar, and E.G. Sirer. Karma: A secure economic framework for Peer-to-Peer resource sharing. In *Proc. of P2PECON*, 2003.

[129] S. Voulgaris, M. Jelasity, and M. van Steen. A robust and scalable Peer-to-Peer gossiping protocol. In *Proc. of AP2PC*, 2003.

[130] K. Walsh and E.G. Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *Proceedings of the 3rd conference on Networked Systems Design & Implementation-Volume 3*, pages 1–1. USENIX Association, 2006.

[131] Wikipedia. Gini coefficient. `http://en.wikipedia.org/wiki/Gini_coefficient`.

[132] Wikipedia. Pareto optimality. `http://en.wikipedia.org/wiki/Pareto_optimality`.

[133] L. Xiong and L. Liu. A reputation-based trust model for peer-to-peer ecommerce communities. 2003.

[134] L. Xiong and L. Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *Knowledge and Data Engineering, IEEE Transactions on*, 16(7):843–857, 2004.

[135] B. Yang and H. Garcia-Molina. PPay: micropayments for peer-to-peer systems. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 300–310. ACM, 2003.

[136] B. Yu and M. Singh. Incentive mechanisms for peer-to-peer systems. *Agents and Peer-to-Peer Computing*, pages 39–58, 2005.

[137] C. Zhang, P. Dhungel, Z. Liu, and K.W. Ross. BitTorrent darknets. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[138] ZNet. Parecon official website. `http://www.parecon.org`.

Summary

Peer-to-Peer System Design: A Socioeconomic Approach

It has gradually become clear that Peer-to-Peer (P2P) systems should not be conceived in the manner of conventional computing systems. Consequently, over the years, ideas from social science in general and economics in particular have made their way to P2P systems to deal with the novel challenges that such systems throw up.

In Chapter 2 of this thesis we examine many of the socioeconomic ideas that have been employed in the literature. We discover that the economic ideas that have been utilized for incentive related works in P2P systems are imbued in the rational action framework. The assumptions entailed in this framework drive the solutions to free-riding in P2P incentive works, and also impinge upon the usage of terms such as fairness, social welfare, etc.

Our examination leads us to accomplish three major tasks: 1) We argue for the reconsideration of commonly used concepts and paradigms; 2) We propose novel socioeconomic concepts and approaches for the design of P2P systems; and 3) We open up new areas that could benefit from the use of socioeconomic ideas.

In Chapter 3 we take inspiration from opinion gathering techniques such as VoxPopuli; voting tools such as the Ballot Box; and general acceptance of the opinion of *experienced* members of the society, to devise a voting infrastructure to fight spam in Tribler, a P2P media distribution system

In Chapter 4 we indicate that private BitTorrent communities that use sharing ratio enforcement for incentivizing contribution from peers, can face problems such as credit crunches. In Chapter 5, we show that such problems are exacerbated when peers do not follow the rational user model. We also present some ways in which these problems can be rectified.

In Chapter 6 and 7, we argue for the consideration of alternate viewpoints viz a viz concepts such as fairness, freeriding and social welfare; and for formulation of alternatives to game-theoretic solution concepts for assessing the robustness and performance of distributed protocols. In Chapter 6 we propose effort based incentives with a novel concept of fairness, for improving efficiency and fairness in P2P systems. In Chapter 7 we argue that the goal of most earlier P2P works, of achieving a Nash equilibrium when all peers are rational, is limited. We present Design Space Analysis, a simulation based method that: allows for flexible user behavior assumptions; aims to analyze protocols

more comprehensively and not just on a single point in the design space; and provides a tournament based approach for evaluating protocol robustness and performance.

Samenwatting

Het ontwerpen van peer-to-peer systemen: een socio-economische benadering

Het is geleidelijk aan duidelijk geworden dat Peer-to-Peer (P2P) systemen niet als conventionele computersystemen moeten worden beschouwd. Als gevolg hiervan hebben ideen vanuit de sociale wetenschappen en de economische wetenschappen hun toepassing gevonden in het onderzoek naar P2P systemen, om hiermee de nieuwe uitdagingen van deze systemen te lijf te gaan.

In hoofdstuk 2 van dit proefschrift verkennen we een groot aantal socio-economische ideen die in de literatuur worden behandeld. We ontdekken dat de economische ideen die worden toegepast binnen het onderzoek naar incentives in P2P systemen uitgaan van het principe van rationele actie. De aannamen die hierop gebaseerd zijn vormen binnen het bestaande onderzoek de basis voor oplossingen voor freeriding, en zijn gestoeld op het gebruik van termen als fairness, social welfare, etcetera.

Het doel van ons onderzoek is drieledig: 1) we beargumenteren een heroverweging van de gebruikelijke concepten en paradigmas; 2) we presenteren nieuwe socio-economische concepten en methoden voor het ontwerpen van P2P systemen; en 3) we inventariseren nieuwe richtingen die voordeel kunnen hebben bij het gebruik van socio-economische ideen.

In hoofdstuk 3 halen we inspiratie uit technieken voor het vergaren van meningen, zoals VoxPopuli; uit stem-methoden zoals de Ballot Box; en uit het principe van de algemene acceptatie van de mening van ervaren leden van een community. Hiermee creren we een stem-infrastructuur om spam te reduceren in Tribler (een P2P mediadistributiesysteem).

In hoofdstuk 4 tonen we aan dat zogenaamde private BitTorrent communities, die gebaseerd zijn op sharing-ratio enforcement, te maken kunnen krijgen met problemen zoals een credit crunch. In hoofdstuk 5 laten we zien dat zulke problemen erger worden wanneer het gedrag van gebruikers niet overeenkomt met het rationele gebruikersmodel. We presenteren tevens een aantal manieren waarop deze problemen verholpen kunnen worden.

In hoofdstukken 6 en 7 beargumenteren we de relevantie van alternatieve perspectieven op concepten zoals fairness, freeriding, en social welfare; en het formuleren van alternatieven voor speltheoretische oplossingsconcepten voor het evalueren van de robu-

ustheid en performance van gedistribueerde protocollen. In hoofdstuk 6 presenteren we effort based incentives met een nieuw concept van fairness, om daarmee de efficintie en fairness in P2P systemen te verbeteren. In hoofdstuk 7 beargumenteren we dat het doel van de meeste eerdere P2P onderzoekenhet bereiken van een Nash-equilibrium wanneer alle peers rationeel zijnbeperkt is. We presenteren Design Space Analysis, een simulatie-methode welke: flexibele aannamen toestaat voor gebruikersgedrag; tot doel heeft om protocollen uitgebreider te analyseren, niet alleen vanuit een enkel punt in de design space; en een methode biedt om de robuustheid en performance van protocollen te evalueren.

Curriculum vitae

Rameez Rahman was born in Karachi, Pakistan in 1980. The greater part of his childhood was spent in Beijing, China (1985-1993). Rameez completed his BS in Computer Science, MSc in Distributed Systems and PhD in Computer Science from *Sir Syed University, Karachi*, *Essex University, UK* and the *Delft University of Technology, The Netherlands*, respectively. Upon the completion of his Masters degree from the UK, Rameez returned to Pakistan and worked in the software industry. In 2007, he left his job as a project manager at Matrix Systems (pvt) Ltd, Karachi to pursue a PhD in Computer Science from Delft University of Technology.

Rameez now intends to join the academia in Pakistan.

Employment

- **Matrix Systems (pvt) Ltd., Karachi.**

  Project Manager and Team Lead, 2006–2007.

  Responsibilities included managing the Web Technologies Team. I also served as an active member in the Software Engineering Process Group and played a key role in the company's drive towards the successful implementation of ISO 9001:2000 and CMMI Level 2.

- **ITA Networks Inc. USA, Karachi Office.**

  Software Engineer, 2005–2006.

  ITA Networks primarily develops software for secure messaging. I worked on improving the company's flagship product, ITA Secure Messaging. The language of development was Java. (www.itanetworks.com)

Publications

- **Design Space Analysis for Modeling Incentives in Distributed Systems.** Rameez Rahman, Tamas Vinko, David Hales, Johan Pouwelse and Henk Sips. *Accepted for publication at ACM SIGCOMM, 2011.*
- **Fast Download but Eternal Seeding: The Reward and Punishment of Sharing Ratio Enforcement.** Adele Lu Jia, Rameez Rahman, Tamas Vinko, Johan Pouwelse and Dick Epema. *Accepted for publication at IEEE P2P, August 31-September 2, 2011, Kyoto, Japan.*
- **No more crash or crunch: Sustainable credit dynamics in a P2P community.** Rameez Rahman, David Hales, Tamas Vinko, Johan Pouwelse and Henk Sips. *In Proc. of International Conference on High Performance Computing & Simulation, HPCS 2010, France, 2010*
- **Improving Efficiency and Fairness in P2P Systems with Effort-Based Incentives.** Rameez Rahman, Michel Meulpolder, David Hales, Johan Pouwelse, Dick Epema and Henk Sips. *In Proc. of IEEE International Conference on Communication 2010.*
- **Modeling and Analyzing the Effects of Firewalls and NATs in P2P Swarming Systems.** Lucia D'Acunto, Michel Meulpolder, Rameez Rahman, Johan Pouwelse and Henk Sips. *In Proc. of IPDPS 2010, pp. 1-8. IEEE Computer Society*
- **BitTorrent or BitCrunch: Evidence of a credit squeeze in BitTorrent?** David Hales, Rameez Rahman, Boxun Zhang, Michel Meulpolder and Johan Pouwelse. *In Proc. of Wetice 2009, pp. 99-104. IEEE CS Press.*
- **Robust vote sampling in a P2P media distribution system.** Rameez Rahman, David Hales, Michel Meulpolder, Vincent Heinink, Johan Pouwelse and H. Sips. *In Proc. of IPDPS 2009, Rome. IEEE Computer Society*

Technical Reports

- **Revisiting Social Welfare in P2P** Rameez Rahman, Michel Meulpolder, David Hales, Johan Pouwelse and Henk Sips. *Technical report PDS-2009-003, Delft University of Technology*
- **Modeling and Analyzing the Effects of Firewalls and NATs in P2P Swarming Systems.** Lucia D'Acunto, Michel Meulpolder, Rameez Rahman, Johan Pouwelse and Henk Sips. *Technical report PDS-2009-004, Delft University of Technology*

Areas of Interest

I am interested in modeling and analyzing complex adaptive systems. Such systems, where individual elements, e.g., agents, follow some fundamental rules and can adapt their behavior by interacting with their environment, hold great fascination for me. Simulation based modeling and analysis of such systems can open up new possibilities of a revolutionary understanding of various facets of human life.

Pedagogical Experience

Served as TA for the 'Introduction to High Performance Computing' course, under Prof. Henk Sips. 2007–2011.

Extra-curricular activities

Written several articles on sociopolitical themes published in various newspapers and magazines including DAWN, The News, Daily News, etc. Also written articles for the Delft university paper, TU Detla.