



QLectives – Socially Intelligent Systems for Quality
Project no. 231200

Instrument: Large-scale integrating project (IP)
Programme: FP7-ICT

Deliverable D2.1.2

Fundamental algorithms for sustaining cooperation in realistic environments

Submission date: 2011-03-01

Start date of project: 2009-03-01

Duration: 48 months

Organisation name of lead contractor for this deliverable: TUD

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document information

1.1 Author(s)

Author	Organisation	E-mail
David Hales	TUD	dave@davidhales.com
Tamás Vinkó	TUD	T.Vinko@tudelft.nl

1.2 Other contributors

Name	Organisation	E-mail
Rameez Rahman	TUD	rrameez@gmail.com
Rahim Delaviz	TUD	rahim.delaviz@gmail.com
Adele Jia	TUD	adele.lu.jia@gmail.com
Nazareno Andrade	TUD	nazareno@gmail.com
Lucia D'Acunto	TUD	L.dAcunto@tudelft.nl
Michel Meulpolder	TUD	meulpolder@gmail.com
Dick Epema	TUD	D.H.J.Epema@tudelft.nl
Henk Sips	TUD	H.J.Sips@tudelft.nl
Johan Pouwelse	TUD	peer2peer@gmail.com
Nigel Gilbert	UniS	n.gilbert@surrey.ac.uk

1.3 Document history

Version#	Date	Change
V0.1	20 December 2010	First draft internal consortium version
V0.2	14 January 2011	Minor corrections and clarifications
V1.0	13 February 2011	Approved version to be submitted to EC
V1.1	24 March 2011	Chapter 2 extended on request of the reviewers

1.4 Document data

Keyworlds	QLectives, peer-to-peer, BitTorrent, reciprocity, indirect reciprocity, inequality, evolution
Editor address data	T.Vinko@tudelft.nl
Delivery date	17 February, 2011

1.5 Distribution list

Date	Issue	E-mail
	Consortium members	QLECTIVES@list.surrey.ac.uk
	Project officer	Jose.FERNANDEZ-VILLACANAS@ec.europa.eu
	EC archive	INFSO-ICT-231200@ec.europa.eu

QLectives Consortium

This document is part of a research project funded by the ICT Programme of the Commission of the European Communities as grant number ICT-2009-231200.

University of Surrey (Coordinator)

Department of Sociology/Centre
for Research in Social Simulation
Guildford GU2 7XH
Surrey
United Kingdom
Contact person: Prof. Nigel Gilbert
E-mail: n.gilbert@surrey.ac.uk

Technical University of Delft

Department of Software Technology
Delft, 2628 CN
Netherlands
Contact Person: Dr Johan Pouwelse
E-mail: j.a.pouwelse@tudelft.nl

ETH Zurich

Chair of Sociology, in particular
Modelling and Simulation
Zurich, CH-8092
Switzerland
Contact person: Prof. Dirk Helbing
E-mail: dhelbing@ethz.ch

University of Szeged

MTA-SZTE Research Group on
Artificial Intelligence
Szeged 6720, Hungary
Contact person: Dr Mark Jelasity
E-mail: jelasity@inf.u-szeged.hu

University of Fribourg

Department of Physics
Fribourg 1700
Switzerland
Contact person: Prof. Yi-Cheng Zhang
E-mail: yi-cheng.zhang@unifr.ch

University of Warsaw

Faculty of Psychology
Warsaw 00927
Poland
Contact Person: Prof. Andrzej Nowak
E-mail: nowak@fau.edu

Centre National de la Recherche Scientifique, CNRS

Paris 75006,
France
Contact person: Dr. Camille ROTH
E-mail: camille.roth@polytechnique.edu

Institut für Rundfunktechnik GmbH

Munich 80939
Germany
Contact person: Dr. Christoph Dosch
E-mail: dosch@irt.de

QLectives introduction

QLectives is a project bringing together top social modelers, peer-to-peer engineers and physicists to design and deploy next generation self-organising socially intelligent information systems. The project aims to combine three recent trends within information systems:

- **Social networks** - in which people link to others over the Internet to gain value and facilitate collaboration
- **Peer production** - in which people collectively produce informational products and experiences without traditional hierarchies or market incentives
- **Peer-to-Peer systems** - in which software clients running on user machines distribute media and other information without a central server or administrative control

QLectives aims to bring these together to form Quality Collectives, i.e. functional decentralised communities that self-organise and self-maintain for the benefit of the people who comprise them. We aim to generate theory at the social level, design algorithms and deploy prototypes targeted towards two application domains:

- **QMedia** - an interactive peer-to-peer media distribution system (including live streaming), providing fully distributed social filtering and recommendation for quality
- **QScience** - a distributed platform for scientists allowing them to locate or form new communities and quality reviewing mechanisms, which are transparent and promote quality

The approach of the QLectives project is unique in that it brings together a highly inter-disciplinary team applied to specific real world problems. The project applies a scientific approach to research by formulating theories, applying them to real systems and then performing detailed measurements of system and user behaviour to validate or modify our theories if necessary. The two applications will be based on two existing user communities comprising several thousand people - so-called "Living labs", media sharing community tribler.org; and the scientific collaboration forum EconoPhysics.

Executive summary

The aim of this deliverable is to identify and translate theoretical models of cooperation formation into algorithms for ICT systems. We draw on the review of models from deliverable D1.1.1, and the proposed applications of those models given in D2.1.1. We have focused on the QMedia application domain in this phase of QLectives work and particularly BitTorrent related issues. The main contributions of this deliverable are:

- Summarise published work which examines potential improvements to a deployed distributed indirect reciprocity mechanism - chapter 1.
- Give an overview of on-going work which applies an evolutionary inspired tournament approach to automatically assess the quality of a space of BitTorrent-like sharing protocols - chapter 2.
- Summarise published work which presents a detailed analysis of the relationship between performance and equality within the direct reciprocity mechanism of BitTorrent - chapter 3

We do not discuss here the implementation of the Channel concept within Tribler (as discussed in the previous deliverable D2.1.1) since this is now an implementation issue and is discussed in deliverable D4.3.2. In addition the BarterCast II implementation, influenced by the work presented in chapter 1, is also discussed in D4.3.2.

Contents

1	Improving Accuracy and Coverage in an Internet-Deployed Reputation Mechanism	1
1.1	Introduction	1
1.2	The BarterCast Reputation Mechanism	2
1.3	The Crawler and Proposed Modifications	5
1.4	Experimental Setup and Results	7
1.5	Summary	9
2	Evolutionary Inspired Distributed Algorithmic Mechanism Design	11
2.1	Introduction	11
2.2	Game-Theoretic Analysis of BitTorrent	13
2.3	Design Space Analysis	19
2.4	Applying DSA to P2P File Swarming Systems	21
2.5	BitTorrent Protocol Space - Some Initial Results	24
2.6	Summary	25
3	BitTorrent's Dilemma: Enhancing Reciprocity or Reducing Inequity	26
3.1	Introduction	26
3.2	A Fluid Model for BitTorrent	27
3.3	Analysis of Four Strategies	31
3.4	Related Work	37
3.5	Summary	37
4	Summary and further research questions	38

Chapter 1

Improving Accuracy and Coverage in an Internet-Deployed Reputation Mechanism

In this section we give a brief overview of results assessing the accuracy and coverage of proposed modifications to the currently deployed distributed indirect reciprocity (reputation) system in Tribler - called BarterCast. Full results and detailed explanations can be found in the associated published paper of which this chapter is a summary only [10].

1.1 Introduction

P2P systems can benefit from *reputation mechanisms* through which peers evaluate the reputations of the participants of the system and are therefore able to identify good service providers. Two central properties of a reputation mechanism are its *accuracy*, that is, how well a peer can approximate "objective" reputation values when calculating the reputation of other peers, and its *coverage*, that is, the fraction of peers for which an interested peer is able to compute reputation values. Inaccurate or partial reputation evaluation may lead to misjudgment, poor behavior, and finally, system degradation. The BarterCast mechanism [28] is an Internet-deployed reputation mechanism that is used by the Tribler Bittorrent-based file-sharing client [33] to select good bartering partners and to prevent free-riding. In this paper we evaluate the accuracy and the coverage of BarterCast, propose three modifications to this mechanism, and evaluate these modifications with respect to accuracy and coverage. The evaluation is performed using empirical data collected by crawling the Tribler P2P network over a 3-month period.

P2P file-sharing systems are characterized by large populations and high turn-over. In such setting, two participants interacting will often have no previous experience with each other, and will be thus unable to estimate each others' behavior in the system. If choosing among potential interaction partners is important, such configuration is an issue. The fundamental idea behind a reputation mechanism is that individual behavior does not usually change radically over time, and past activity is a good pre-

dictor of future actions [38]. Using this idea, a reputation mechanism collects information on the past behavior of the participants in a system and quantifies these information into reputation values. In a distributed reputation mechanism, depending on how the information about peers' behavior are disseminated or how the reputation values are computed, each participant may have different reputation values for the same participants.

We have previously designed and implemented the BarterCast reputation mechanism in our Bittorent-based P2P client Tribler. In BarterCast, peers exchange messages about their upload and download actions, and use the collected information to calculate reputations. From the BarterCast messages it receives, each peer builds a local weighted, directed graph with nodes representing peers and with edges representing amounts of transferred data. This subjective graph is then used by each peer to calculate the reputation values of other peers by applying the maxflow algorithm to the graph, interpreting the edge weights as "flows."

In this paper we propose three modifications to the BarterCast reputation mechanism, and we evaluate the accuracy and the coverage of the original BarterCast reputation mechanism and of all combination of these three modifications. First, rather than have each peer execute the maxflow algorithm to compute reputations from its own perspective, we make each peer do so from the perspective of the node with the highest *betweenness centrality* [11] in its subjective graph. The second modification consists in using a gossiping protocol that fully disseminates the BarterCast records in the whole system rather than limiting the exchange of these records to one hop. In the third modification we increase the maximal path length in the maxflow algorithm to 4 or 6 instead of 2 as in the original BarterCast. In order to evaluate the original BarterCast reputation mechanism and our three modifications, we have crawled the Tribler P2P system for 83 days to obtain as many BarterCast records of the Tribler peers as possible. From the records obtained from each peer, we emulate its reputation computations by reconstructing its *subjective view*, represented by the subjective graph of the peer (in this paper the terms subjective graph and subjective view are synonyms). We then use this graph to execute the maxflow algorithm with and without modifications.

In the rest of the paper, we first explain the BarterCast mechanism in detail and we define the metrics accuracy and coverage. In Section 1.3, we explain the crawler and the data collecting process, and we describe the collected data. In Section 1.3, we state the problem with the current version of BarterCast and explain the modifications we propose. In Section 1.4, first we explain the experimental setup and then the experimental results are presented.

1.2 The BarterCast Reputation Mechanism

In this section, we first explain the BarterCast mechanism in detail and then we formulate the metrics accuracy and coverage in this mechanism.

1.2.1 The BarterCast Mechanism

The BarterCast mechanism is used by the Tribler Bittorrent client to rank peers according to their upload and download behavior. In this mechanism, a peer whose upload is much higher than its download gets a high reputation, and other peers give a high priority to it when selecting a bartering partner. In BarterCast, when two peers exchange content, they both log the amount of transferred data and the identity of the exchange partner in a BarterCast record; these records store the total cumulative amounts of data transferred in both directions since the first data exchange between the peers. In BarterCast, each peer regularly contacts other peers in order to exchange BarterCast records. Peer sampling for selecting to whom to send BarterCast records is done through a gossip protocol called BuddyCast, which is at the basis of Tribler. In BuddyCast, peers regularly contact each other in order to exchange lists of known peers and content.

Using the BarterCast message exchange mechanism, each peer creates its own current local view of the upload and download activity in the system. Formally, the receiver of BarterCast records creates and gradually expands its *subjective graph*. The subjective graph of peer i is $G_i = (V_i, E, \omega)$, where V_i is the set of nodes representing the peers about whose activity i has heard through BarterCast records, and E is the set of weighted directed edges (u, v, w) , with u and $v \in V_i$ and w the total amount of data transferred from u to v . Upon reception of a BarterCast record (u, v, w) , peer i either adds (a) new node(s) and a new edge to its subjective graph if it did not know u and/or v , or only (a) new directed edge(s) if it did know u and v but did not know about the data transfer activity between them, or adapts the weight(s) of the existing edge(s) between u and v . If peer i receives two BarterCast records with the same sender u and the same receiver v from different peers, it keeps the record that indicates lower amounts of data transferred in order to avoid invalid reports from malicious peers that try to inflate their uploads. Furthermore, the direct experience of the peer has higher priority than received reports from others.

In order to calculate the reputation of an arbitrary peer $j \in V_i$ at some time, peer i applies the maxflow algorithm [44] to its current subjective graph to find the maximal flow from itself to j and vice versa. Maxflow is a classic algorithm in graph theory for finding the maximal flow from a source node to a destination node in a weighted graph. When applying maxflow to the subjective graph, we interpret the weights of the edges, which represent amounts of data transferred, as flows. The original maxflow algorithm by Ford-Fulkerson [44] tries all possible paths from the source to the destination, but in BarterCast, in order to limit the computation overhead, only paths of length at most 2 are considered. The rationale for expecting that this limit is sufficient is that the majority of peers may have indirect relationships through popular intermediaries [31], in which case using two hops in maxflow provides sufficient data for the evaluation of reputations. Using the values $F_2(\cdot, \cdot)$ as computed with the 2-hops maxflow algorithm, the subjective reputation of peer j from peer i 's point of view is calculated as:

$$S_{ij} = \frac{\arctan(F_2(j, i) - F_2(i, j))}{\pi/2}, \quad (1.1)$$

and so $S_{ij} \in [-1, +1]$. If the destination node j is more than two hops away from i ,

then its reputation is set to 0.

In Figure 1.1 a simple subjective graph is shown in which peer i as the owner of the graph evaluates the reputation of peer j . In this graph, $F_2(i, j) = 11$ and $F_2(j, i) = 5$, and so $R_i(j) = -0.89$.

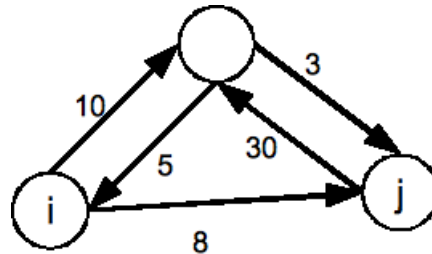


Figure 1.1: A sample subjective graph.

Using a flow algorithm (e.g., maxflow in BarterCast) is like doing a collaborative inference where the knowledge of all involved nodes is included in the computation of the final value. Beside this, flow algorithms like maxflow are more resilient against sybil attacks - in which a single peer uses multiple identities to boost its reputation - than trivial operations like averaging or summation are not [8]. The BarterCast mechanism can be generalized in the form of *flow-based* mechanisms. Such mechanisms have two common features. First, the relation between participants is shown as a graph. Second, there is a function ϕ which calculates the flow of a specified metric from a node set I to a destination node set J , and the obtained flow is used to calculate the final reputation value.

1.2.2 Accuracy and Coverage

As the term *accuracy* indicates, it is a measure of how close an estimated reputation value is to an "objective" or real value. In a distributed mechanism like BarterCast, depending on how the feedback records are disseminated, peers may have different opinions about the reputation of a peer at the same time. Each peer also at each point in time has an *objective reputation* value, O_j , that is calculable only if the evaluator peer has a global view of the activity of all peers. In our case, only the crawler has such a view and using the collected data we can calculate the objective reputations. If U_j and D_j are the total upload and download by peer j , then its objective reputation is

$$O_j = \frac{\arctan(U_j - D_j)}{\pi/2} \quad (1.2)$$

Using the objective and subjective reputations, the estimation error is defined as the absolute value of the difference between the subjective and objective values:

$$e(i, j) = \text{abs}(S_{ij} - O_j) \quad (1.3)$$

Higher estimation errors mean lower accuracy and vice versa.

Coverage is another important metric that expresses how well a node is located and can reach other nodes in the graph. Denoting by $F_h(\cdot, \cdot)$ the maximum flow computed with the maxflow algorithm using all paths of length less than or equal to h , in the subjective graph G the h -hop coverage of node i is defined as

$$c_G(i, h) = |\{u | F_h(i, u) > 0 \text{ or } F_h(u, i) > 0\}| \quad (1.4)$$

So the coverage of node i in a graph is the number of nodes at a distance at most h from node i with non-zero maximum flow to or from i . Dividing the coverage by the number of nodes normalizes it into the interval of $[0, 1]$ and makes it possible to compare this metric in graphs of different size.

1.2.3 Related Work

The BarterCast mechanism was designed by Meulpolder et al. to distinguish free-riders and cooperative peers in file-sharing environments. After the first release, Seuken et al. [41] proposed an improvement to make it more resilient against misreporting attacks. Their solution is based on ignoring some of the feedback reports. Also, this solution could cut down the severity of the attack, but on the other hand it increases the feedback sparsity. Xiong et al. [45] show that the feedback sparsity is an issue in large distributed systems, and that a lack of enough feedback can lead to lower accuracy and coverage.

Besides BarterCast, several other distributed reputation mechanisms have been proposed for P2P systems, but they use different methods to calculate reputation values. EigenTrust [19] is based on summation of direct observations and indirect data and uses centralized *matrix operations* to compute the left eigen vector. The CORE system [6] uses *arithmetic weighted averaging* on historical data to calculate reputation values. The BarterCast mechanism best fits in a class of mechanisms which use flow-based reputation functions as defined by Cheng et al. [8].

1.3 The Crawler and Proposed Modifications

To collect the required dataset consisting of the BarterCast records of all (or at least, many) Tribler peers for analysis, we have crawled the Tribler network for 83 days, from June 20 until September 9, 2009. Except for some slight differences, the crawler works as an ordinary Tribler client. Discovery of the new peers is done through the BuddyCast protocol, which is the gossiping engine of the Tribler client. When a new peer is discovered with this protocol, it is added to a list. The crawler hourly contacts all peers in this list and asks them for their latest BarterCast records by including the timestamp of the latest record it does have of each peer. Using the BarterCast records received by the crawler from each peer, we can reconstruct the subjective graph of that peer in the same way the peer builds it.

The discovered peers have different ages, some of them having been installed and running for months and others just for a few days or even hours. So, when the crawler asks a peer for BarterCast records for the first time, it might receive very old records that are useless because they correspond to peers that were online in the past but no

longer participate in the system. To mitigate this problem, when the crawler contacts a peer for the first time, it uses the start time of the crawl, that is, 00:00 hours on June 20, 2009, so that the discovered peers will only include BarterCast records fresher than the crawl start time in their replies.

Another problem in doing the crawling is the size of the reply messages. If a peer is asked for all its records at once, the reply message might be large and sending it may be problematic. To prevent this intrusive effect in the crawling, in each contact, peers are only asked for 50 records that they have not sent already. Because of a potentially high churn rate, this limitation causes a side effect and for some of the peers that go offline the crawler is unable to fetch all their records. To have a reliable analysis, such incomplete views should be removed. Because in each contact a peer is limited to send at most 50 records, it is probable that, having a multiple of 50 records from a peer means that it has not sent all its records. As a consequence, to filter out incomplete views, all views of the size of a multiple of 50 are removed.

To be able to sort the collected records and to account for the time difference with remote peers, the crawler asks peers to send their local time as well. When the crawler receives such information, it logs the remote peer's time and its own local time. Using these two times and the timestamp of the record (available in the record payload) the collected records can be sorted. If t_p and t_c denote the local time of the remote peer and the crawler, respectively, and t_r is the record timestamp, then the relative record occurrence time is:

$$t_c - t_p + t_r \quad (1.5)$$

This relative time is used in the experiments to sort the BarterCast records.

During the crawling period, the crawler collected 547,761 BarterCast records from 2,675 different peers. After filtering out the incomplete views, 416,061 records were left, collected from 1,442 peers, which means that although 46% of the views are incomplete, they contain only 24% of the collected records. All the subsequent processing and analysis in this paper is based only on complete views.

An analysis of the collected data set shows that the accuracy and the coverage of the current BarterCast mechanism are low and need to be improved. The mean of the estimation error is 0.664, which is the same as the average difference between two random values in the interval of possible reputation values, $[-1, +1]$. This means that a random guess for the subjective reputation value has the same precision as using the BarterCast mechanism. Similarly, the coverage of the BarterCast mechanism is very low at 0.032. In order to remedy this situation, we propose the following three modifications to the BarterCast mechanism.

1.3.1 Modification 1: Using Betweenness Centrality

Betweenness centrality was introduced by Freeman [11] as a measure of the number of shortest paths passing through a node. In a graph $G = (V, E)$, if δ_{st} is the number of shortest paths between two arbitrary nodes s, t of G , and $\delta_{st}(v)$ is the number of these paths that pass through node v , then the betweenness centrality of node v is $\beta(v) = \sum_{s \neq v \neq t} \frac{\delta_{st}(v)}{\delta_{st}}$. A higher betweenness centrality means a higher participation of the node in connecting other nodes, and also a higher flow that passes through it.

Another feature of this measure is that in contrast to connectivity (the sum of in and out degrees of a node), which is a local quantity, betweenness centrality is a quantity across the whole graph; nodes with many connections may have a low betweenness centrality and vice versa [4]. Betweenness centrality has been used in the analysis of various topics, like transportation, social networks, and biological networks, but to the best of our knowledge it has not been used in reputation systems.

In the original BarterCast mechanism, a peer i as the owner of the subjective graph G_i , in evaluating the reputation of peer j , runs the maxflow algorithm to compute the maximum flow from itself to j and from j to itself. In the proposed modification, first, node i finds the node with the highest betweenness centrality in G_i , and then replaces itself with that node in the maxflow execution. By this change, the evaluator peer benefits from the centrality feature of the central node and uses the collected data in a better way.

1.3.2 Modification 2: Using Full Gossip

The second modification is obtained by changing the way BarterCast records are disseminated. In the original version, peers only use *1-hop* message passing and they are not allowed to forward the received records. Peers only report their own download and upload activities to the peers that are discovered by the BuddyCast protocol. This method limits the effect of misreporting but it is not efficient in spreading the BarterCast records. Specially if a peer goes offline, its upload and download activity are not disseminated, and when it comes online again, very few peers know about its activities. In this modification, instead of using 1-hop message passing, we assume that there is a *full gossiping* protocol that spreads records without the hop limitation, so that in principle all online peers eventually receive all propagated records.

1.3.3 Modification 3: Lifting the Maxflow Hop-Count Restriction

In the third modification we lift the restriction of 2 on the hop count in the maxflow algorithm and increase it to 4 or 6 hops. With this change, more nodes are involved in the maxflow algorithm and the chance of reaching a node, and so increasing the coverage, is increased.

1.4 Experimental Setup and Results

In this section we give a brief overview of results assessing the accuracy and coverage of the proposed modifications. Full results and explanations can be found in the published technical paper of which this chapter is a summary only [10]. To calculate the experimental results we emulate the creation of subjective graphs using the BarterCast records received by the crawler, and we emulate their computation of the reputation values of those peers to which they appear to have uploaded data. Then we present the experimental results and compare the effect of the proposed modifications on accuracy and coverage.

1.4.1 Coverage

The bar chart in Figure 1.2 shows the number of covered peers for all combinations of the proposed modifications. It is expected that only existing peers can be covered by the evaluator peers, and so in all of our experiments the maximum possible value for the coverage is 123 (the number of existing peers as obtained from the crawler data). The left half of the graph shows the cases in which the central node is used in the maxflow algorithm and the right half the view owner itself. As the graph shows, full gossiping boosts the coverage dramatically. Using the central node increases the coverage too, specially in 2-hops maxflow, but for a larger number of hops, it is less effective. Increasing the number of hops has more or less the same influence as using the central node, and in both dissemination methods the biggest improvement is seen when we go from 2 to 4 hops.

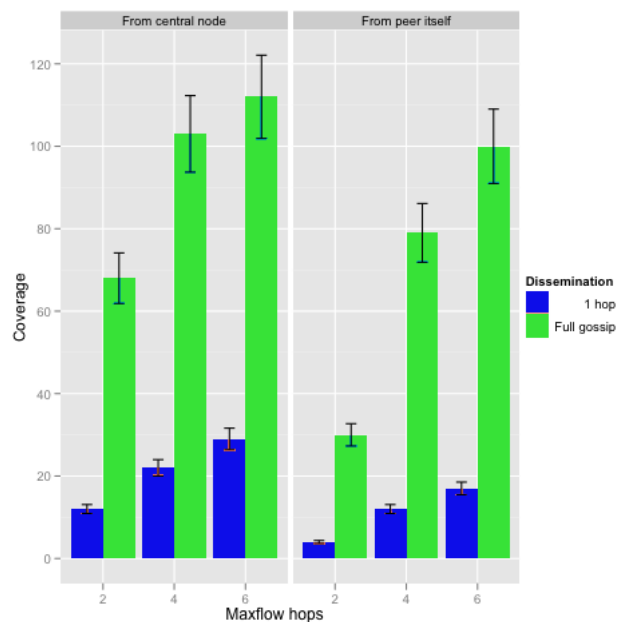


Figure 1.2: The coverage of the BarterCast mechanism in different scenarios. (Error bars show the standard error of mean.)

1.4.2 Accuracy

In Figure 1.3 we show the fractions of nodes for which either the central node in the subjective graph or the local peer provides a better estimation of the reputation value for different numbers of hops in maxflow and in both 1-hop and full-gossip dissemination. In practice, equal reputation estimation means that both reputation values are equal to 0. As the left hand of the figure (1-hop dissemination) shows, in more than 80% of the cases the central node and the view owner give the same estimation. When we move to full gossiping, the situation changes considerably, and using the central node gives better estimations. Especially with 4 and 6 hops, the number of cases for

which the central node is better is twice the number of cases for which the view owner is better.

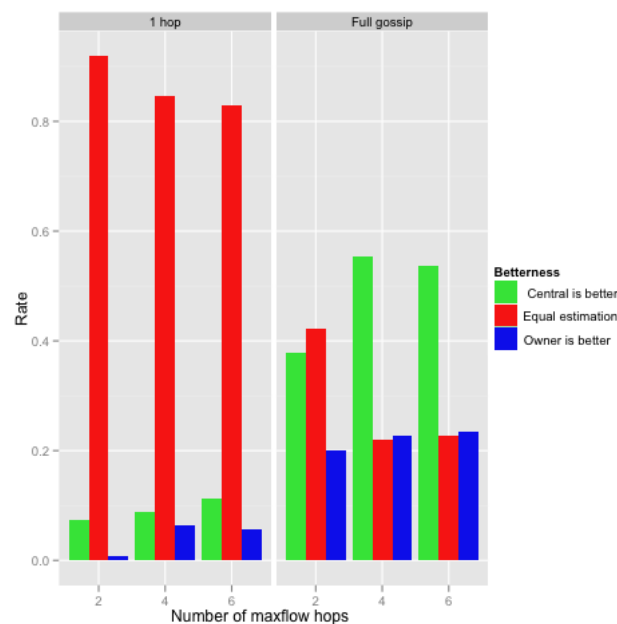


Figure 1.3: Comparing the accuracy of the central node against the view owner in the BarterCast mechanism.

Figure 1.3 only shows which combination of the methods is better, but it does not tell how much they are better. To have a grasp of the improvement rate we compare the mean and the median of estimation errors. Figure 1.4 shows the mean and its standard error for all combinations of the modifications. As the graph shows, only changing the number of hops or using the central node does not improve much, and using the full gossiping is needed. Then, using both the central node and a higher number of hops decrease the estimation error, and when all modifications are applied, the mean of the errors becomes 0.404.

1.5 Summary

In this chapter we overviewed results from an associated paper [10] in which we performed an empirical analysis of the accuracy and the coverage of the BarterCast reputation mechanism and proposed three applicable modifications to improve these values: using betweenness centrality, using full gossip instead of 1-hop dissemination of BarterCast records, and increasing the path length in the maxflow algorithm. Our results show that using full gossip leads to the largest improvement according to our metrics. The other two modifications provide significant improvements, but only if combined with full gossip.

After understanding the improvements leveraged by changes in the design of BarterCast, some open questions related to the proposed improvements need now to be addressed. Also full gossiping increases the dissemination performance, but it is more

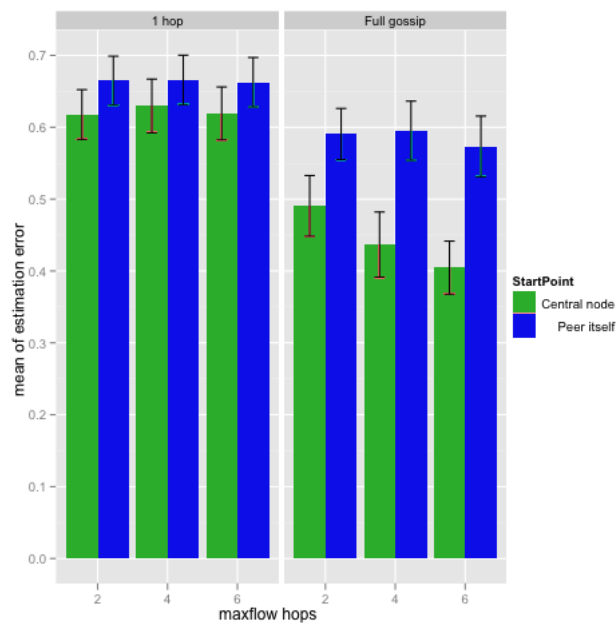


Figure 1.4: The mean of the estimation error in the BarterCast mechanism. (Error bars show the standard error of mean.)

vulnerable to misreporting attacks, and the indirect reports should be treated carefully. A possible solution for this problem could be to put the indirect reports in a secondary view and to add them to the primary view, used for reputation evaluation, if they are received from more than a certain number of peers or by highly reputed peers. Another method to address the misreporting attack is the use of double signatures. In this solution, before disseminating a record, the content sender and receiver sign the associated BarterCast record using their private keys. Using this technique no other peer can eavesdrop and change the record.

Chapter 2

Evolutionary Inspired Distributed Algorithmic Mechanism Design

In this chapter we summarise ongoing work (yet to be published) which applies a evolutionary inspired approach to designing new P2P protocols by creating a strategy space of protocols and performing round-robin tournaments between them in realistic simulated settings. This approach is directly inspired by the famous tournaments conducted by Axelrod and described in the classic book on the evolution of cooperation [2]. Currently we have not applied full evolution within the strategy space but may perform such experiments in future work.

Traditionally, P2P designers have used game theory for modeling and analyzing incentives. We argue that traditional techniques focus on single points in the strategy space, employ unrealistic definitions of robustness, and also do not address populations where nodes may not behave rationally. We apply a game theoretic analysis to a popular P2P protocol and design a Nash equilibrium variant. However, we discover the limitations of this analysis upon testing it with a new model and simulation based methodological approach. Our new approach relies on searching for desired protocols in a general design space. Using this approach designers can obtain meaningful measures of performance and robustness.

2.1 Introduction

Incentives play an important role 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 [29]. 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 [7, 39, 42]. However, little attention has been paid to exploring new approaches that can be equally generally applied, and which can overcome limitations of game-theoretic analysis.

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

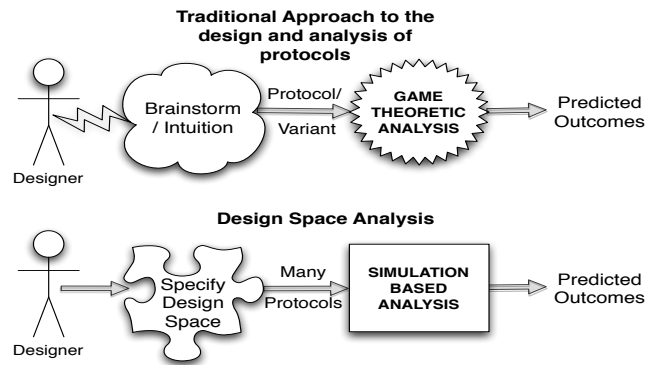


Figure 2.1: Complementary approaches for the analysis and design of protocols.

intractable [27]. 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.

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 [2], 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 2.1. 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

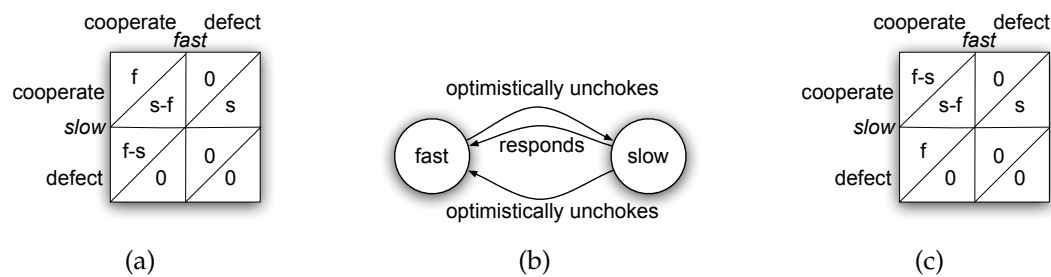


Figure 2.2: Analysis of the BitTorrent Dilemma: (a) The payoffs of the BT Dilemma for slow and fast peers; (b) An abstract illustration of interaction between slow and fast peers; (c) Modified BT payoffs in view of slow peers' opportunity costs.

detailed analysis of protocols. The results of DSA can be used to gain insights into the workings of protocols and for designing deployable systems.

2.2 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 2.1 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 [5, 18, 23, 30].

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 [29]. Then we present an analysis of this model for multiple bandwidth classes. Under our assumptions, which are different from previous work [35], 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.

2.2.1 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 itera-

tions. We do not model 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 2.2(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. 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 [12].

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. Figure 2.2(b) depicts this scenario. 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* [37]. For simplicity, we refer to it here as the *BitTorrent Dilemma*.

We note here that Levin et al. [23] 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 2.2.3. 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 2.2.3.

Next we give an analytical model of BitTorrent for multiple bandwidth classes, using the BitTorrent Dilemma game as depicted in Figure 2.2(a).

2.2.2 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 2.2(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

Table 2.1: 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)
$E^r[X \rightarrow c]$	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.
$E[X \rightarrow c]$	the expected number of 'free game wins' that peer c obtains from class $X \in \{A, B, C\}$.
N_r	$N_A + N_B + N_C - U_r - 1$

TFT, cooperate unconditionally, thereby giving c a *free game win*.

We use the notation summarized in Table 2.1. 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.

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 2.2(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 classes is N_C/N_r , giving $E[A \rightarrow C] = N_A \times N_C/N_r$, 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 $E[A \rightarrow c] = N_A/N_r$.

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

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

$$E^r[C \rightarrow c] = U_r - E[A \rightarrow c] - K, \quad (2.1)$$

where $K = 1 - ((1 - E[A \rightarrow c])(1 - \frac{1}{U_r}))^{U_r}$. Finally, the number of peers in contention for 'free game wins' by peer c in the same class is $N_C - 1 - E^r[C \rightarrow c]$, which gives $E[C \rightarrow c] = (N_C - 1 - E^r[C \rightarrow c])/N_r$.

2.2.3 Is BitTorrent TFT a Nash equilibrium?

Under certain assumptions, it was shown in [35] 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 (2.1), we find that BT is not a Nash equilibrium. In our model, the payoff structure in BitTorrent can be modified to devise a strategy that can do better than BitTorrent, thereby showing that BitTorrent is not a Nash equilibrium.

Next, we discuss how we devise a protocol improvement called *Birds*¹ 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 2.2(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 2.2(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 2.2(c) so that the dominant strategy of both slow and fast peers is to defect against each other.

Analytically, this leads to $E_B^r[A \rightarrow c] = E_B^r[B \rightarrow c] = 0$, and obviously $E_B^r[C \rightarrow c] = U_r$. For the 'free game wins', there is no change as compared to BitTorrent. For the interactions of peers in the same class, we find that we can use the same argument as used for BitTorrent, thus $E_B[C \rightarrow c] = (N_C - 1 - U_r)/N_r$.

In the following we give a proof of Birds being a Nash equilibrium and BitTorrent not being a Nash equilibrium. We use the notation introduced in Table 2.1 and the results from Sections 2.2.2 and 2.2.3.

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 $E_B^r[B \rightarrow c]' = N_B/N_r$, 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

¹Birds of a feather stick together. Using this protocol peers try to stick with others from their own class.

from other peers in this class will be $E_B^r[C - c]' = U_r - K$ for Birds and

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

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

$$E_B[C \rightarrow c]' > E[C \rightarrow c]'$$

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

$$\begin{aligned} E_B[C \rightarrow c]' &= \frac{N_{C'}}{N_C}(N_C - E^r[C \rightarrow c]')/N_r, \\ E[C \rightarrow c]' &= E_B[C \rightarrow c]' + \frac{N_C - E_B^r[C \rightarrow c]'}{N_C N_r}, \end{aligned}$$

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

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

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

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

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

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

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

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

and

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

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

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 2.2(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 [40].

2.2.4 Discussion

We have applied the 'Traditional Approach' depicted in Figure 2.1 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 [35] 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 [35], that questions about BitTorrent's robustness were raised. Locher et al. [24] 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 [30]. 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? Furthermore, game-theoretic analyses, for tractability purposes, usually *do* require high level of abstraction. Therefore, it is likely that a single analysis will not encompass many relevant and important details for modeling complex protocols.

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.

2.3 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.

2.3.1 Key elements of DSA

We consider the elements that are integral to Design Space Analysis. We compare and contrast each of these with the corresponding elements in traditional game-theoretic analysis.

Flexible behavioral assumptions. In DSA, we relax behavioral assumptions. Specifically, unlike traditional game-theoretic analysis, we do not limit ourselves to the rational framework, where nodes are supposed to be self-interested. By foregoing the assumptions entailed in this framework, we consider a great variety of protocols, which may not necessarily be rational. Protocols may, in the words of Axelrod [2], “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* [20] 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.

In comparison, often in game-theoretic analysis only a single point in the design space is taken into consideration, e.g., peer selection in BitTorrent.

An example of specifying a design space, with both the parameterization and actualization phases, will be described in detail in Sections 2.4.1 and 2.4.2 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 predict their effects. Heuristic based approaches can provide partial solutions relatively fast, however, without any guarantees on the level of goodness of the measures.

Game theoretic analyses generally use solution concepts, which involve finding the equilibrium strategy. Such concepts usually require high levels of abstractions and can become intractable when applied to the large design space for complex protocols.

2.3.2 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.

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 tradeoff 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 the PRA space; formally, we define a function $\mathcal{S} : D \rightarrow [0, 1]^3$, 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 with all other protocols. 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.

2.4 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.

2.4.1 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 stranger policy. 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.

2.4.2 Actualization of a Specific P2P Protocol Design Space

We define some specific actualizations of a BitTorrent-like file swarming system, as described in Section 2.2, based on the general design space of Section 2.4.1. 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 [2, 14, 32]. 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 [15].
- 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 [2].

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 [32].
- I5) *Sort Loyal*, ranks peers in order of those who have cooperated with the peer for the longest durations. This has been inspired by [14].
- I6) *Random*, does not rank peers and chooses them randomly. This has been inspired by [22].

After applying the ranking function, a peer chooses the k top peers. 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 stranger 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 [23].
- 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 explore². Our purpose here is to show the practicality of the DSA analysis by analyzing a considerable space of unique protocols.

²For instance, we do not consider variants for resource allocation to strangers. Also, we do not consider Peer Discovery.

2.4.3 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 2.4.2.

Simulation Model

We use a cycle-based simulation model, in which time consists of *rounds*. 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 2.4.2. We run our simulation experiments with 50 peers, which is a good approximation of an average BitTorrent swarm-size [13]. 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. [30]. We assume that all peers always have data that others are interested in.

Methodology

Based on the PRA quantification, as described in Section 2.3.2, 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 75-25, where 75% of the peers follow protocol Π , while 25% execute other protocols in the space, and observe similar results. We do 10 runs for each particular encounter between two protocols. 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.

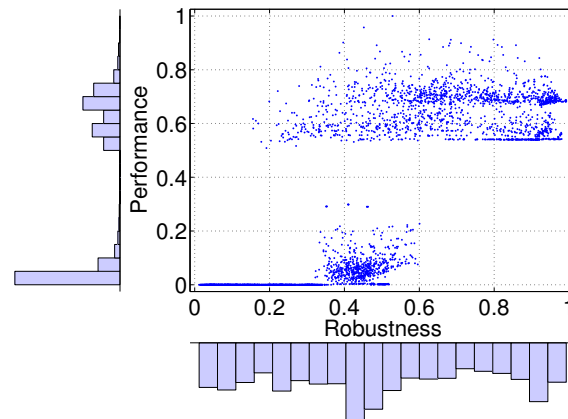


Figure 2.3: Scatter plot of all 3270 protocols in the design space with Robustness against Performance. The results presented here are a synthesis of over 107 million individual simulation runs. Histograms are also shown.

Next, we present results of applying the PRA quantification over the design space³ described in Section 2.4.2.

2.5 BitTorrent Protocol Space - Some Initial Results

Figure 2.3 shows all the 3270 protocols actualized in Section 2.4.2, with their normalized Robustness and Performance values. Given the methodology of conducting the PRA quantification as described in Section 2.4.3, 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 2.4.3. We determined the variance in the runs for each protocols' performance and robustness values and found it to be very low.

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 2.3, 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 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.

³We note here that the entire series of simulations was computationally tractable.

The detailed discussion of Figure 2.3 and further experiments with our DSA approach is still an ongoing work and will be reported in a research paper as well as in the next year's deliverable of Stream 2.

2.6 Summary

In this chapter we have presented a game-theoretical analysis of BitTorrent and a protocol improvement which was inspired by this analysis. Then we have summarised on-going work that applies an approach, which we consider quite general, for exploring a space of distributed system protocol variants. Detailed descriptions, analysis and further experiments with performance and robustness of protocol variants in controlled environments (emulations) will be reported in a soon to be produced technical paper.

Chapter 3

BitTorrent's Dilemma: Enhancing Reciprocity or Reducing Inequity

This chapter gives an overview of a published technical paper. Further details can be found in the technical paper [16].

Enhancing reciprocity has been one of the primary motivations for the design of incentive policies in BitTorrent-like P2P systems. Reciprocity implies that peers need to contribute their bandwidth to other peers if they want to receive bandwidth in return. However, the over-provisioning that characterizes today's BitTorrent communities and the development of many next-generation P2P systems with real-time constraints (e.g., for live and on-demand streaming) suggest that more effort can be devoted to reducing the inequity (i.e., the difference of service received) among peers, rather than only enhancing reciprocity. Inspired by this observation, in this work we analyze in detail several incentive mechanisms that are used in BitTorrent systems, and explore several strategies that influence the balance between reciprocity and equity. Our study shows that (i) reducing inequity leads to a better overall system performance, and (ii) the behavior of seeders (i.e., peers that hold a complete copy of the file and upload it for free) influences whether reciprocity is enhanced or inequity reduced.

3.1 Introduction

BitTorrent is a popular peer-to-peer (P2P) protocol for file distribution over the Internet. In order to induce cooperation among peers, BitTorrent incorporates an incentive mechanism based on direct *reciprocity*, where nodes prefer uploading to peers who have contributed to them in the past at the highest speeds. This incentive mechanism was designed to allow peers to obtain their file of interest even in *resource-constrained* scenarios, e.g., when only a few peers exist that hold a complete copy of the file (*seeders*, in BitTorrent terminology), or during flash-crowds.

However, the *BitTorrent ecosystem* is nowadays extremely diverse. For example, a recent measurement study [26] has shown that most BitTorrent communities are over-provisioned, i.e., there are significantly more seeders than downloaders. Also, the design of many next-generation P2P systems, such as those for the distribution of live and on-demand streaming [1], [17], has been inspired by the BitTorrent paradigm.

The real-time constraints of these systems require that all peers are provided with a certain minimum download speed (in order to support the bitrate of the video) and that peers do not earn more utility in downloading at rates much faster than that. These observations suggest that it is not necessary to always enhance reciprocity; in some cases it is more advisable to reduce *inequity* among peers, instead. One of the first studies of this trade-off in BitTorrent-like systems was provided by Fan *et al.* [3].

In this chapter, we extend earlier work by introducing a more detailed model and analyzing *how* the incentive mechanism of the BitTorrent protocol can be tuned to enhance reciprocity or reduce inequity. Furthermore, in our study we consider the implications of exchanging BitTorrent's standard incentive mechanism with one that is based on effort rather than speed. Finally, we also analyze the role of the seeders. Hence, we provide significant insights into the implications of this important trade-off. Our contributions can be summarized as follows:

- we provide an analytical model that characterizes the inherent relationship between a peer's performance and the design parameters of the BitTorrent protocol that are responsible for its incentive mechanism (Section 3.2).
- we use this model to analyze different strategies to enhance reciprocity, reduce inequity and understand the role of the seeders (Section 3.3).
- we consider the impact of these strategies on the overall system performance (Section 3.3).

Overall, our work aids in informing the design choices that best fit the requirements of a BitTorrent-like P2P system.

3.2 A Fluid Model for BitTorrent

In this section we first introduce the basics of the BitTorrent protocol which are relevant for our work, then we present our model, and finally we illustrate its validation by means of a discrete-event simulator.

BitTorrent Overview:

Incentive policies play a key role in BitTorrent-like systems, as they determine how peers distribute their limited upload bandwidth to other peers. BitTorrent's original incentive policy is *tit-for-tat* (TFT), in which a peer favors other peers that have recently reciprocated at the highest rate. More specifically, every peer has a number of upload slots available, which are divided into two categories, *regular unchoke slots* and *optimistic unchoke slots*. Downloaders (referred to as *leechers*, in BitTorrent terminology) choose which peers will be allocated to regular unchoke slots according to TFT. On the contrary, peers to be allocated to optimistic unchoke slots are chosen randomly from the neighbors set. While regular unchoke slots are used to enhance reciprocity, optimistic unchoke slots serve the purpose of 1) potentially discovering new faster peers and 2) allowing new peers to bootstrap (i.e., obtain their first pieces of the file).

BitTorrent systems also include special peers called seeders, who have a complete copy of the file and share it without any direct benefit to do so. Two popular seeding policies are: 1) *favoring fast peers* (FF): seeders allocate their regular upload slots to peers that downloaded at the fastest rates and optimistic unchoke slots randomly; 2) *random seeding* (RS): seeders have no preference and just choose peers randomly.

Notation	Definition
F	the size of the file shared in the swarm.
μ_i	the upload capacity of a peer in class i .
D_i	the download capacity of a peer in class i .
d_i	the per connection download capacity of a peer in class i .
u_i	number of unchoke slots opened by a peer in class i , $u_i^{(reg)}$ and $u_i^{(op)}$ for regular and optimistic unchoke slot.
x_i	number of leechers in class i .
π_i	fraction of leechers in class i , $\pi_i = x_i / \sum_i x_i$.
y_i	number of seeders in class i .
λ_i	the arrival rate of leechers in class i .
γ_i	the rate at which seeders in class i leave the system.
α_{ij}	the number of upload slots allocated by a leecher in class i to a leecher in class j .
β_{ij}	the number of upload slots allocated by a seeder in class i to a leecher in class j .
n_i	the number of download slots opened by a class i leecher
U_{ij}	the total upload bandwidth allocated from class i to class j .
D_{ij}	the fraction of upload capacity of leechers in class i allocated to leechers in class j .
S_{ij}	the fraction of upload capacity of seeders in class i allocated to leechers in class j .

Table 3.1: Notation of our BitTorrent model

Model description:

We follow a similar fluid modeling approach as Qiu *et al.* [34] and Meulpolder *et al.* [25]. The notation we use is shown in Table 3.1. Similar to the approach in [25], we group peers into different classes according to their upload capacities, but we introduce the notion of *per connection download capacity*. For each class i , the evolution of the number of leechers, $x_i(t)$, and the number of seeders, $y_i(t)$, is as follows:

$$\begin{aligned}
 \frac{dx_i(t)}{dt} &= \lambda_i - \frac{\sum_j U_{ji}(t)}{F}, \\
 \frac{dy_i(t)}{dt} &= \frac{\sum_j U_{ji}(t)}{F} - \gamma_i y_i(t).
 \end{aligned}
 \tag{3.1}$$

In a steady state, although peers are arriving and departing, the total system population is constant. So it holds that $\frac{dx_i(t)}{dt} = \frac{dy_i(t)}{dt} \equiv 0$, which implies:

$$\lambda_i F = \gamma_i y_i F = \sum_j U_{ji} = \sum_j (D_{ji} x_j + S_{ji} y_j) \mu_j. \quad (3.2)$$

Combining this with Little's Law ($x_i = \lambda_i T_i$), the average download speed for leechers in class i can be calculated as:

$$\frac{F}{T_i} = \frac{F \lambda_i}{x_i} = \frac{1}{x_i} \sum_j (D_{ji} x_j + S_{ji} y_j) \mu_j. \quad (3.3)$$

We discuss how to derive the upload bandwidth allocation (D_{ji} and S_{ji} respectively) in the following subsection.

Bandwidth allocation:

Without loss of generality, we assume that $\mu_1 < \mu_2 < \dots < \mu_N$ and $D_1 < D_2 < \dots < D_N$.

Leechers utilize the TFT policy. As an indirect result, high capacity peers only unchoke low capacity peers using optimistic unchoke slots:

$$\alpha_{ij} = u_i^{(op)} \pi_j \quad i, j = 1, 2, \dots, N, i > j. \quad (3.4)$$

Due to their faster upload speed, higher-capacity leechers will get reciprocated when they upload to lower-capacity leechers. On average, each leecher in class j should reciprocate $(\alpha_{ij} x_i) / x_j = u_i^{(op)} \pi_i$ leechers in class i , as long as it has enough upload slots. In case there are not enough upload slots, leechers in higher classes are reciprocated first, i.e.:

$$\alpha_{ji} = \min\{u_i^{(op)} \pi_i, u_j^{(reg)} - \sum_{i < p \leq N} \alpha_{jp}\} + u_j^{(op)} \pi_i. \quad (3.5)$$

Seeders do not need to be reciprocated since they only upload altruistically. For seeders who adopt the FF policy we have:

$$\begin{aligned} \beta_{iN} &= u_i^{(reg)} + u_i^{(op)} \pi_N, \\ \beta_{ij} &= u_i^{(op)} \pi_j \quad \forall i, j \text{ and } j < N, \end{aligned} \quad (3.6)$$

while for seeders who adopt the RS policy it holds:

$$\beta_{ij} = u_i \pi_j. \quad (3.7)$$

BitTorrent uses TCP as transport layer protocol. TCP specifies that a peer's upload (download) capacity is equally divided over all connections, unless some of the connections have a bottleneck. When such a bottleneck exists, normally the leftover bandwidth is equally divided over other connections with a higher link capacity. Taking this into account and the fact that, in a steady state, peers in the same class receive a similar service, the average number of download connections and the per connection download capacity for a peer in class i can be calculated as:

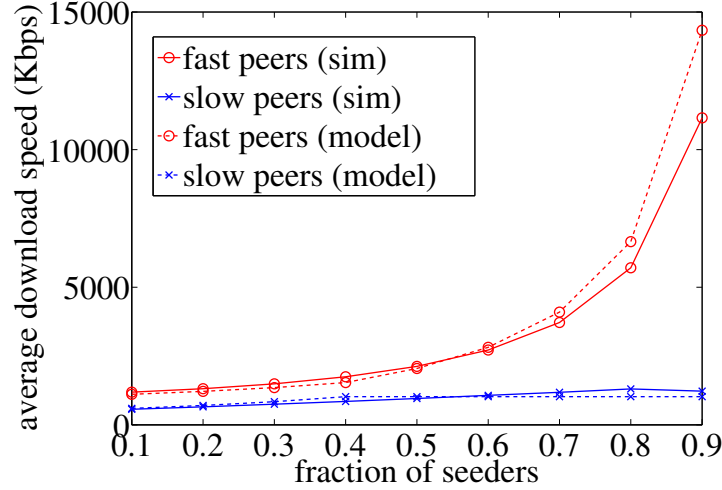


Figure 3.1: The average download speeds of fast and slow peers in a system with 50 fast peers and 50 slow peers, for different fraction of seeders. The capacities of peers are the following: 1024 Kbps up and ∞ down for fast peers; 512 Kbps up and 1024 Kbps down for slow peers. Seeders use the FF policy.

$$n_i = \frac{\sum_{1 \leq j \leq N} \alpha_{ji} x_j + \beta_{ji} y_j}{x_i}, \quad (3.8)$$

$$d_i = \frac{D_i}{n_i}.$$

We now reorder the leechers according to d_i , and we assume that $d_1 < d_2 < \dots < d_N$. The bandwidth allocation can be calculated as:

$$D_{ij} = \frac{\min\left\{\frac{\mu_i(1 - \sum_{p < j} D_{ip})}{\sum_{k \geq j} \alpha_{ik}}, d_j\right\} \cdot \alpha_{ij}}{\mu_i}. \quad (3.9)$$

Replacing D_{ij} , α_{ij} with S_{ij} , β_{ij} respectively, we can calculate a seeder's upload bandwidth allocation in a similar way.

Model Validation:

We have validated our model by means of extensive simulations using a discrete-event simulator that accurately emulates the behavior of BitTorrent at the level of piece transfers. Fig. 3.1 illustrates the simulation results against the model predictions for a system with two classes of peers, fast and slow, from which we can make the following observations:

1. the model predictions are close to the simulation results;
2. the average download speed of both fast and slow peers increases when there are more seeders;
3. the model predictions become less accurate as the fraction of seeders grows. This can be explained considering that, when a high fraction of peers are seeders (above 70

% in this case), fast leechers have a hard time in finding other fast leechers to reciprocate with. While in our model we assume that, in a steady state, leechers can always find enough other leechers.

Currently we have not validated the model against real-world data, such as that collected from the QMedia living lab. This could be a topic for future work.

3.3 Analysis of Four Strategies

In this section, we analyze the balance between enhancing reciprocity or reducing inequity in BitTorrent. Based on our model, we evaluate the following candidate strategies:

- A) fast peers opening more regular unchoke slots;
- B) all peers opening more optimistic unchoke slots;
- C) replacing TFT with an effort-based incentive policy;
- D) seeders' role: favoring fast peers vs seeding randomly.

We use the following performance metrics:

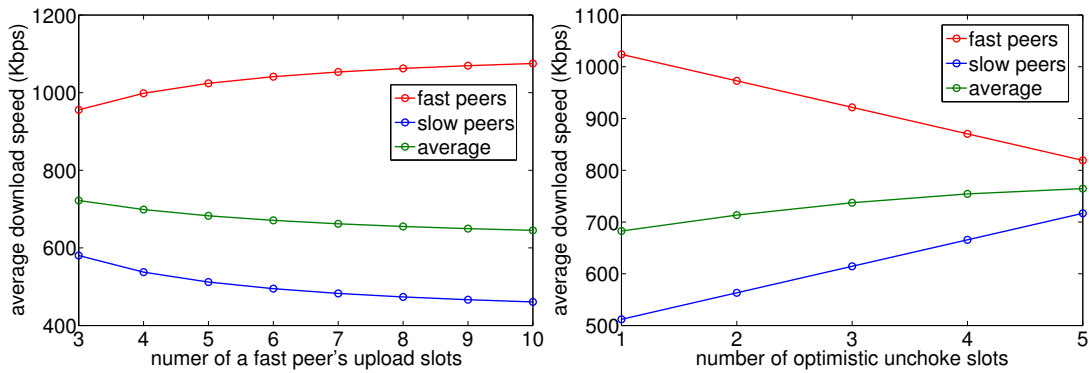
1. *download speed*: we use this metric to characterize performance;
2. *sharing ratio*: the ratio between the total amount of data uploaded and downloaded; this metric represents fairness in relation to contribution to the system (e.g., a sharing ratio close to 1 for all peers means that all peers have contributed as much data as they have consumed);
3. *inequity coefficient*: the largest download speed divided by the smallest download speed; it indicates fairness in relation to the bandwidth capacity that peers receive from the system.

Unless stated otherwise, we consider a system with two classes of peers, fast (1024 Kbps up and ∞ down) and slow (512 Kbps up and 1024 down).

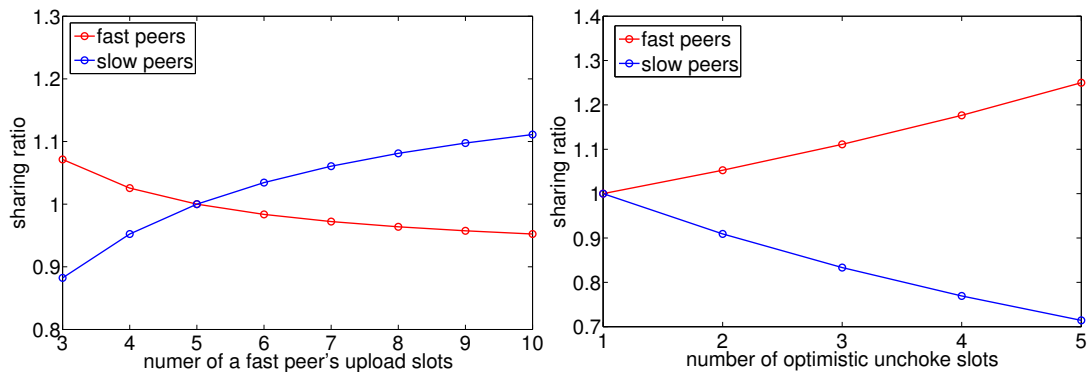
Strategy 1: enhancing reciprocity with fast leechers opening more regular slots

Regardless of a peer's class, opening more upload slots can help a peer to 1) find more potential fast peers, or to 2) weaken another peer's potential monopoly on its uploading bandwidth since less bandwidth will be allocated to each upload slot. On the other hand, opening too many slots is neither realistic nor reasonable, since too many TCP connections could deteriorate link performance. Also it would become harder for slow peers to succeed in competing for reciprocity with faster peers.

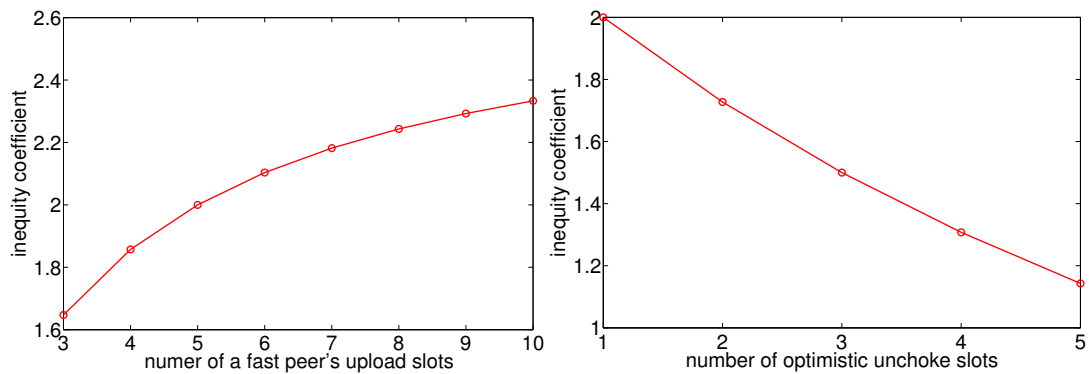
Given the above considerations, fast peers have a stronger motivation to open more slots than slow peers, since they may benefit from more extensive exploration, while remaining competitive in TFT. Having fast peers open more upload slots is a way to enhance reciprocity, as more bandwidth will be allocated to the regular unchoke slots. Fig. 3.2(a) shows that as the number of upload slots of fast peers increases, their download speed improves (we can observe a growth of 10% when the number of slots goes from 3 to 10), while the average download speed of all peers decreases (10% with the



(a) Average download speed



(b) Sharing ratio



(c) Inequity coefficient

Figure 3.2: The influence of the number of upload slot in a system with 100 leechers and no seeders.

number of slots from 3 to 10). This is due to the increasing inequity (almost 50%) between the two classes of peers, as shown in Fig. 3.2(c). On the other hand, we notice that the sharing ratio of fast peers decreases as they open more slots, and that of slow peers increases (Fig. 3.2(b)). The perfect reciprocity (sharing ratio equal to 1 for both fast and slow peers) is achieved when fast peers open 5 upload slots.

In the following theorem we state the conditions necessary to achieve the perfect reciprocity.

Theorem. *In a BitTorrent system with two classes of peers, no seeders, and no download bottleneck, perfect reciprocity is achieved if and only if:*

$$\frac{\mu_f u_s}{\mu_s u_f} = \frac{u_f^{(op)} + u_s^{(op)}}{u_f^{(op)}}. \quad (3.10)$$

Proof. We first show that for a system with perfect reciprocity, Eq. 3.10 holds. The sharing ratio of a leecher in class i in a steady state is equal to the ratio of its upload and download speed, i.e.:

$$\frac{\mu_i}{\lambda_i F / x_i} = \frac{\mu_i x_i}{F \sum_{j \in \{f,s\}} D_{ji} x_j \mu_j}. \quad (3.11)$$

Perfect reciprocity implies that leechers in different classes achieve the same sharing ratio, i.e.:

$$\frac{\mu_f x_f}{\sum_{j \in \{f,s\}} D_{jf} x_j \mu_j} = \frac{\mu_s x_s}{\sum_{j \in \{f,s\}} D_{js} x_j \mu_j}. \quad (3.12)$$

It follows that Eq. 3.10 holds.

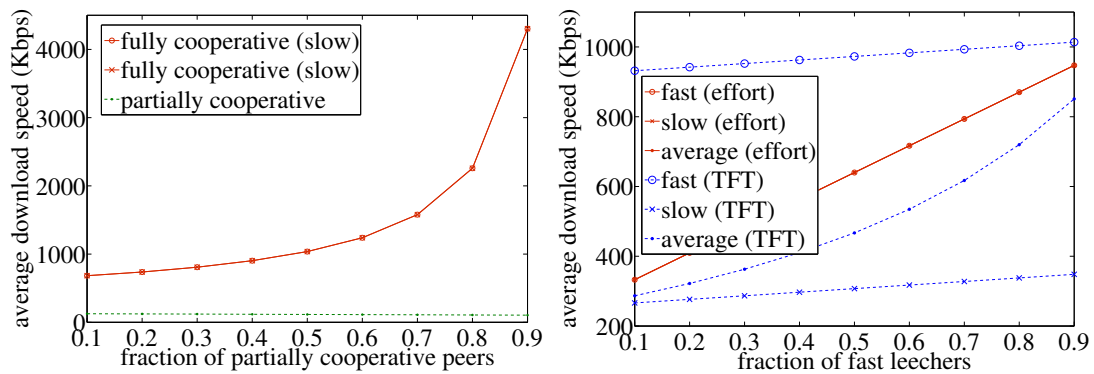
Next we show that when Eq. 3.10 holds, perfect reciprocity is achieved. Substituting Eq. 3.10 into Eq. 3.11, we get Eq. 3.12, which implies that fast and slow leechers have the same sharing ratio. It follows that perfect reciprocity is achieved. \square

From the above theorem it follows that, when we use $\mu_f = 1024$, $\mu_s = 512$, $u_s = 5$ and $u_s^{(op)} = u_f^{(op)} = 1$, a perfect reciprocity is obtained for $u_f = u_s = 5$.

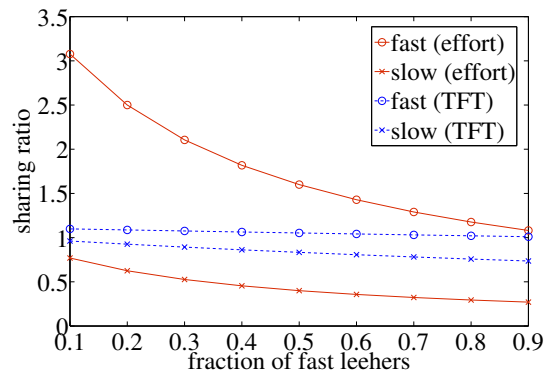
Strategy 2: reducing inequity with leechers opening more optimistic unchoke slots

Here we analyze the influence of having all peers open more optimistic unchoke slots. While peers always open 5 unchoke slots in total, we let the number of their optimistic unchoke slots vary from 1 to 5. As we can see in Fig. 3.2(a), in this way the download speed of slow leechers is improved by 40%, at the expense of the fast leechers. Interestingly, the average download speed of the whole population increases of 15%. Moreover, we observe a 45% decrease of the inequity coefficient (Fig. 3.2(b)).

However, it should be noted that by having peers open more optimistic unchoke slots, the effectiveness of TFT is reduced, as a peer that does not contribute is chosen with the same probability as a cooperative slow peer.



(a) Influence of the fraction of partially cooperative peers (b) Comparison between effort-based and TFT: average download speed



(c) Comparison between effort-based and TFT: sharing ratio

Figure 3.3: The performance of effort-based mechanism. The fast and slow peers' upload capacities are 1024 and 256 Kbps, respectively.

Strategy 3: reducing inequity by replacing TFT with effort-based incentives

Rahman *et al.* [36] have recently proposed a novel incentive mechanism based on effort, rather than speed. More specifically, peers are not rewarded based on the absolute amount of data they provided, but based on the relative amount of bandwidth they make available (utilized or not). With this approach, a slow peer offering all its bandwidth to the system is preferred over a fast peer offering 0.9 of its total bandwidth.

Consider that there are two types of peers in the system, peers that contribute all their upload bandwidth (*fully cooperative*) and peers that only contribute a fraction of it (*partially cooperative*). Let n_p represent the number of partially cooperative peers, and n_{ff} , n_{fs} represent the number of fully cooperative peers that have a low or high upload capacity respectively. Each peer reciprocates fully cooperative peers by allocating regular unchoke slots to them, and punishes partially cooperative peers by only optimistically unchokeing them. The slot allocation for each class of peers can be calculated as:

$$\begin{aligned}
 \alpha_{i(p)} &= \frac{u_i^{(op)} n_p}{n_{ff} + n_{fs} + n_p} \\
 \alpha_{i(ff)} &= \frac{(u_i - \alpha_{i(p)}) n_{ff}}{n_{ff} + n_{fs}} \\
 \alpha_{i(fs)} &= \frac{(u_i - \alpha_{i(p)}) n_{fs}}{n_{ff} + n_{fs}} \quad \forall i \in \{p, ff, fs\}.
 \end{aligned} \tag{3.13}$$

Given Eq. 3.13, the upload bandwidth allocation can be calculated in a similar way as in our earlier analysis.

The idea of this incentive scheme is to reduce inequity among the fully cooperative peers while still punishing the partially cooperative peers. Its effectiveness can be observed in Fig. 3.3(a). In a system where all peers are fully cooperative, the effort-based scheme eliminates the system's inequity and achieves a better overall performance. The average download speed using effort-based incentives is always higher than when using TFT (see Fig. 3.3(b)). Furthermore, the effort-based mechanism leads to a more equal sharing ratio between fast and slow peers (see Fig. 3.3(c)).

Strategy 4: enhancing reciprocity or reducing equity with a seeder's policies

The mainline BitTorrent client has been implemented with two different seeding strategies in different releases. One is the favoring of fast peers. This strategy accelerates a fast leecher's ability to finish downloading, thereby potentially having it serve as fast seeder in the system sooner. The other strategy is seeding randomly. The first strategy is resource-constrained oriented, as it aims at increasing the serving capacity quickly. The second strategy is more equity oriented, as all peers are treated in the same way.

We have applied our model to analyze and compare these two strategies. Fig. 3.4(a) and Fig. 3.4(c) show that if seeders seed randomly, the system achieves a better overall

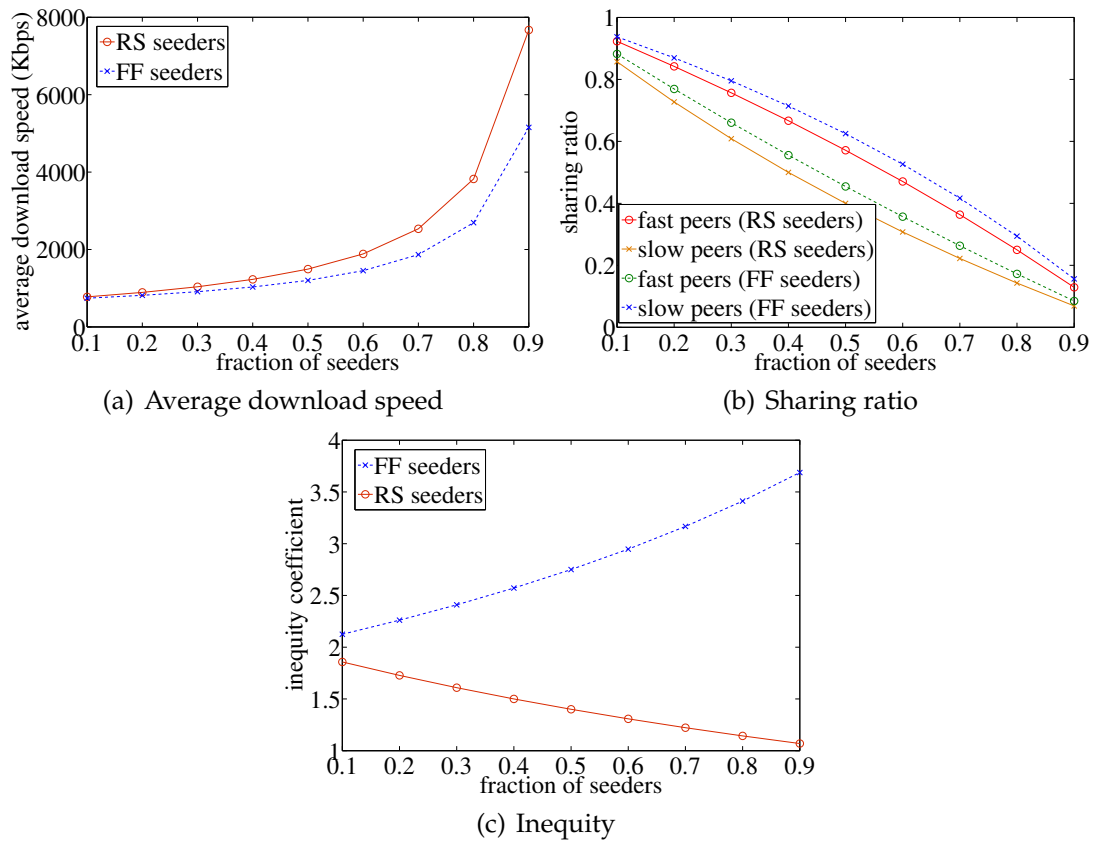


Figure 3.4: The influence of the seeding strategy: favoring fast peers (FF) or randomly seeding (RS)

performance (in terms of a higher average download speed) and the inequity is reduced. On the contrary, if seeders favor fast peers, the reciprocity is enhanced: both fast and slow peers have a sharing ratio higher than in a system where seeders adopt random seeding (Fig. 3.4(b)).

3.4 Related Work

There are a number of studies on modeling and improving BitTorrent's incentive policies. Some earlier work focuses only on homogeneous systems [21], [46], [34]. In [43], the authors consider heterogeneous BitTorrent systems, but only with two classes of peers. Fan *et al.* [3] have developed a general heterogeneous model to evaluate the tradeoff between performance and fairness. Meulpolder *et al.* [25] and Chow *et al.* [9] also provide models for heterogeneous BitTorrent systems, with which they analyze the clustering and data distribution in BitTorrent swarms. While these works all focus on a particular design, we analyze the performance of different incentive policies from a higher level: we consider different BitTorrent applications and stress that merely enhancing reciprocity is not sufficient in the design of a good incentive policy. We furthermore identify several strategies that can be used to enhance reciprocity or reduce inequity.

3.5 Summary

In this chapter, we have provided an overview of an analytical model for heterogeneous BitTorrent systems that captures the essence of BitTorrent's incentive policy. Detailed results are given in the paper this chapter is based on [16]. Based on our model, we have analyzed how TFT could enhance reciprocity or reduce inequity by carefully tuning the number of regular and optimistic unchoke slots. We have also compared TFT to an effort-based incentive policy, and showed that a policy that focuses on reducing inequity leads to a BitTorrent system that achieves a better overall performance. Finally, we have analyzed different seeding policies and our results show that, although seeders do not need to be reciprocated, they can still be used to further enhance reciprocity or reduce inequity among leechers.

Chapter 4

Summary and further research questions

Each of the three lines of research presented in this deliverable raises further research questions. In chapter 1 it was observed that a full gossiping approach can dramatically improve the currently deployed distributed reputation system within Tribler. But this raises issues of security as, in general, gossip is hard to secure - yet solutions have been discussed and could be developed in future work. The current updated BarterCast II implementation is discussed in deliverable D4.3.2. In chapter 3 detailed analysis indicates that both performance and inequality can be improved by simple changes to the resource allocation policy in BitTorrent (based on slot numbers) - although it is an open issue if such a deployed client would spread. To answer this would require measurements and the development of realistic user models. Finally, chapter 2 applies a tournament approach to searching a space of P2P protocols using large-scale simulation results - allowing for realistic assessments of robustness and performance. This latter approach, although applied here to BitTorrent protocol variants, is presented as a general approach that could be applied to a wide range of distributed protocols - which could complement current approaches based on game theoretic analysis of single points in the design space. Future work here could include deploying found variants to test their similarity to the simulation results and also applying the method to a different P2P applications.

Bibliography

- [1] A. Vlavianos, M. Iliofotou and M. Faloutsos. BiToS: Enhancing BitTorrent for Supporting Streaming Applications. In *Proceeding of IEEE Global Internet Symposium*, 2006.
- [2] R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.
- [3] B. Fan, D. M. Chiu and J. C. Lui. The delicate tradeoffs in bittorrent-like file sharing protocol design. In *Proceedings of IEEE ICNP*, 2006.
- [4] M. Barthélemy. Betweenness centrality in large complex networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):163–168, 2004.
- [5] A. Bharambe, C. Herley, and V. Padmanabhan. Analyzing and improving a BitTorrent network’s performance mechanisms. In *INFOCOM*, 2006.
- [6] S. Buchegger and J. Le Boudec. A robust reputation system for mobile ad-hoc networks. *Proceedings of P2PEcon, June*, 2004.
- [7] C. Buragohain, D. Agrawal, and S. Suri. A game theoretic framework for incentives in P2P systems. In *P2P 2003*, pages 48–56, 2003.
- [8] A. Cheng and E. Friedman. Sybilproof reputation mechanisms. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, page 132. ACM, 2005.
- [9] A. Chow, L. Golubchik, and V. Misra. Bittorrent: an extensible heterogeneous model. In *INFOCOM 2009, IEEE*, pages 585–593. IEEE, 2009.
- [10] R. Delaviz, N. Andrade, and J. Pouwelse. Improving accuracy and coverage in an internet-deployed reputation mechanism. In *Proc. IEEE Int’l Conf. Peer-to-peer Comput. (P2P’2010)*, pages 1–9.
- [11] L. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.
- [12] R. Hahnel and A. Library. *The ABCs of political economy: A modern approach*. Pluto Press, 2002.
- [13] T. Hoßfeld, F. Lehrieder, D. Hock, S. Oechsner, Z. Despotovic, W. Kellerer, and M. 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.

- [14] D. Hruschka and J. Henrich. Friendship, cliquishness, and the emergence of cooperation. *Journal of Theoretical Biology*, 239(1):1–15, 2006.
- [15] R. Izhak-Ratzin. Collaboration in BitTorrent systems. In *IFIP-TC Networking*, pages 338–351, 2009.
- [16] A. Jia, L. D’Acunto, M. Meulpolder, J. Pouwelse, and D. Epema. Bittorrent’s dilemma: Enhancing reciprocity or reducing inequity. In *Proc. IEEE Consumer Communications and Networking Conference (CCNC ’11)*, 2011.
- [17] 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 *Proceeding of SPIE MMCN*, 2008.
- [18] S. Jun and M. Ahamad. Incentives in BitTorrent Induce Free Riding. 2005.
- [19] S. Kamvar, M. 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 New York, NY, USA, 2003.
- [20] A.-M. Kermarrec and M. van Steen, editors. *ACM SIGOPS Operating Systems Review 41, Special Issue on Gossip-Based Networking*. 2007.
- [21] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding and X. Zhang. Measurements, analysis, and modeling of bittorrent-like systems. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, 2005.
- [22] B. Leong, Y. Wang, S. Wen, C. Carbunaru, Y. 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.
- [23] D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee. Bittorrent is an auction: analyzing and improving bittorrent’s incentives. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication, SIGCOMM ’08*, pages 243–254, New York, NY, USA, 2008. ACM.
- [24] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free riding in BitTorrent is cheap. In *HotNets-V*, 2006.
- [25] M. Meulpolder, J.A. Pouwelse, D.H.J. Epema and H.J. Sips. Modeling and Analysis of Bandwidth-Inhomogeneous Swarms in BitTorrent. In *9th International Conference on P2P Systems (IEEE P2P ’09)*, 2009.
- [26] M. Meulpolder, L. D’Acunto, M. Capota, M. Wojciechowski, J.A. Pouwelse, D.H.J. Epema and H.J. Sips. Public and private bittorrent communities: A measurement study. In *IPTPS*, 2010.
- [27] G. J. Mailath. Do people play nash equilibrium? lessons from evolutionary game theory. *Journal of Economic Literature*, 36(3):1347–1374, September 1998.

- [28] M. Meulpolder, J. Pouwelse, D. Epema, and H. Sips. BarterCast: A practical approach to prevent lazy freeriding in P2P networks. 2009.
- [29] M. Osborne and A. Rubinstein. *A course in game theory*. The MIT press, 1994.
- [30] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in BitTorrent. In *NSDI*, 2007.
- [31] M. Piatek, T. Isdal, A. Krishnamurthy, and T. Anderson. One hop reputations for peer to peer file sharing workloads. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 1–14. USENIX Association, 2008.
- [32] M. Posch. Win-Stay, Lose-Shift Strategies for Repeated Games–Memory Length, Aspiration Levels and Noise. *Journal of Theoretical Biology*, 198:183–195, 1999.
- [33] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Rein-
ders, M. Van Steen, and H. Sips. Tribler: A social-based peer-to-peer system. *Concurrency and Computation–Practice and Experience*, 20(2):127–138, 2008.
- [34] D. Qiu and R. Srikant. Modeling and performance analysis of bit torrent-like peer-to-peer networks. In *ACM SIGCOMM*, 2004.
- [35] D. Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. *ACM SIGCOMM Computer Communication Review*, 34(4):367–378, 2004.
- [36] R. Rahman, M. Meulpolder, D. Hales, J.A. Pouwelse, D.H.J. Epema and H.J. Sips. Improving efficiency and fairness in p2p systems with effort-based incentives. In *Proceedings of ICC*, 2010.
- [37] E. Rasmusen. *Games and information: An introduction to game theory*. Wiley-Blackwell, 2007.
- [38] P. Resnick and R. Zeckhauser. Trust among strangers in Internet transactions: Empirical analysis of eBay’s reputation system. *Advances in Applied Microeconomics: A Research Annual*, 11:127–157, 2002.
- [39] T. Roughgarden and E. Tardos. How bad is selfish routing. *Journal of the ACM*, 49:236–259, 2002.
- [40] K. Rzdca, A. Datta, and S. Buchegger. Replica placement in p2p storage: Complexity and game theoretic analyses. In *2010 International Conference on Distributed Computing Systems*, pages 599–609. IEEE, 2010.
- [41] S. Seuken, J. Tang, and D. C. Parkes. Accounting Mechanisms for Distributed Work Systems. In *Proceedings 24th AAAI Conference on Artificial Intelligence (AAAI ’10)*, 2010.
- [42] S. Shenker. Making greed work in networks: A game-theoretic analysis of switch service disciplines. *IEEE/ACM Transactions on Networking*, 3:819–831, 2002.

- [43] W. C. Liao, F. Papadopoulos and K. Psounis. Performance analysis of bittorrent-like systems with heterogeneous users. In *PERFORMANCE '07: Proceedings of the 26th International Symposium on Computer Performance, Modeling, Measurements, and Evaluation*, 2007.
- [44] Wikipedia. Ford-fulkerson maxflow algorithm. http://en.wikipedia.org/wiki/Ford-Fulkerson_algorithm.
- [45] L. Xiong, L. Liu, and M. Ahamad. Countering feedback sparsity and manipulation in reputation systems. In *Proceedings of the 2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 203–212. Citeseer, 2007.
- [46] Y. Tian, D. Wu and K. W. Ng. Modeling, analysis and improvement for bittorrent-like file sharing networks. In *Proceedings of INFOCOM*, 2006.