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

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
Tamás Vinkó	TUD	T.Vinko@tudelft.nl
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	20-12-2010	First draft internal consortium version
V0.2	14-12-2010	Minor corrections and clarifications
V1.0	date	Approved version to be submitted to EU

1.4 Document data

Keyworlds	QLectives, peer-to-peer, BitTorrent, reciprocity, indirect reciprocity, inequality, evolution
Editor address data	David Hales
Delivery date	DD MM, YY

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

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

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

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

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

Centre National de la Recherche Scientifique, CNRS
Paris 75006,
France
Contact person: Dr. Camille ROTH
E-mail: camille.roth@polytechnique.edu

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

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 co-operation 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

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 [9].

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 [21] is an Internet-deployed reputation mechanism that is used by the Tribler Bittorent-based file-sharing client [23] 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 turnover. 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 mecha-

nism is that individual behavior does not usually change radically over time, and past activity is a good predictor of future actions [26]. 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* [13] 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 Bittorent 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 [31] 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 [31] 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 [22], in which case using two hops in maxflow provides sufficient data for the evaluation of reputations. Using the values $F_2(.,.)$ as computed with the 2-hops maxflow algorithm,

the subjective reputation of peer j from peer i's point of view is calculated as:

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

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 $R_i(j) = -0.89$.

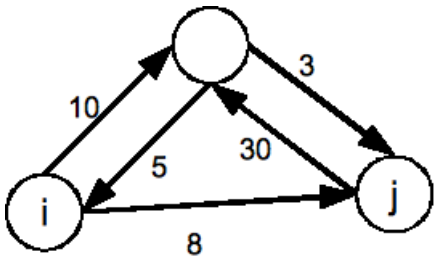


Figure 1.1: A sample subjective graph.

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 it's reputation - than trivial operations like averaging or summation are not [7]. 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

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

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

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

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 $F_h(.,.)$ 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

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

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. [27] 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. [32] 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 [16] 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. [7].

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:

$$t_c - t_p + t_r \tag{1.5}$$

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 [13] 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 [5]. 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 [9]. 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 barchart 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.

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. Espe-

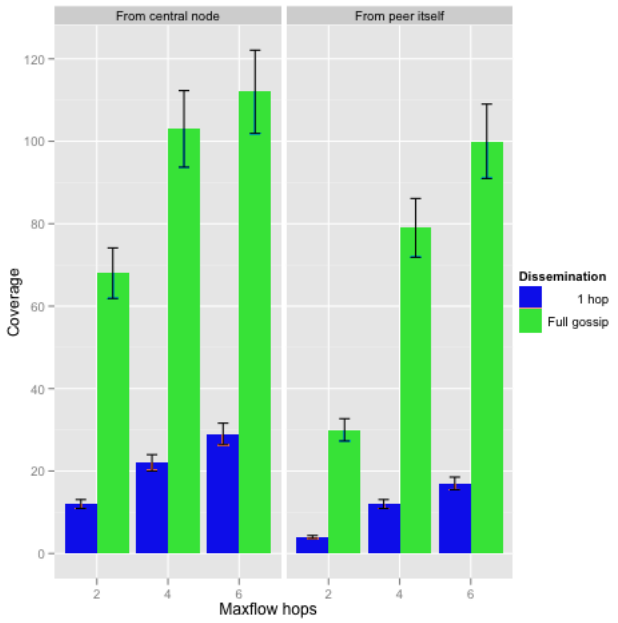


Figure 1.2: The coverage of the BarterCast mechanism in different scenarios. (Error bars show the standard error of mean.)

cially 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 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 [9] 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

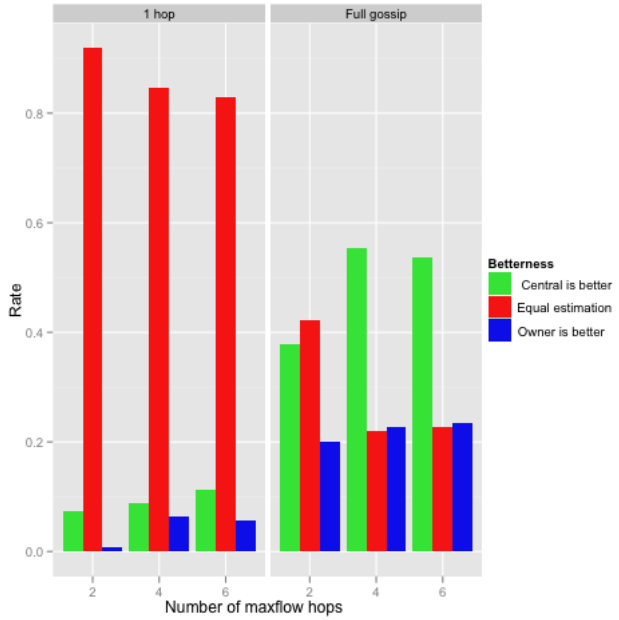


Figure 1.3: Comparing the accuracy of the central node against the view owner in the BarterCast mechanism.

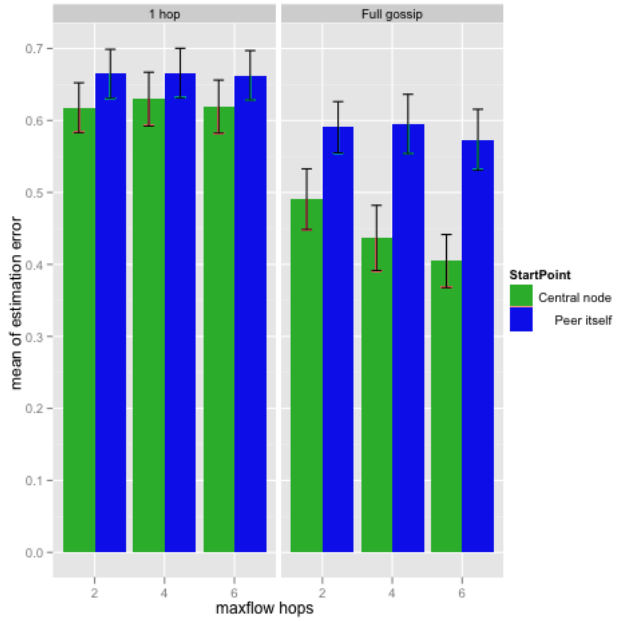


Figure 1.4: The mean of the estimation error in the BarterCast mechanism. (Error bars show the standard error of mean.)

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 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 [3]. 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 for cooperation in peer-to-peer (P2P) networks has been a much studied topic since the early days of P2P literature [29], [28]. The most commonly used tool for the study of incentives in P2P Networks has been Game Theory, the branch of applied mathematics that attempts to capture individual behavior in strategic situations, or games. In this paper, we want to analyze some fundamental questions that we believe have great bearing on protocol analysis and design. Protocol design in the face of many uncertain variables, such as user behavior and a priori unknown adversarial designs, ultimately finds impetus in the designer's intuitions and assumptions. After the protocol has been designed, a game theo-

retic analysis is often applied to see whether it is a Nash equilibrium. In game theory, a Nash equilibrium is a solution concept of a game involving two or more players, in which each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing only his or her own strategy unilaterally [1]. In this paper we address the following fundamental questions: Is game theory is a good tool to gain insights into protocol analysis and design; what are the limitations of game theoretic analysis of P2P Systems; and finally, can an alternate tool be provided that can better guide designers in protocol analysis and design?

BitTorrent as a strategy in iterated games

To address this question, firstly, we consider one of the most popular P2P protocols, BitTorrent. To date no comprehensive analytical model has been put forward which treats BitTorrent as a strategy in an Iterated Prisoner's Dilemma (IPD) between peers having inhomogeneous resources and that also captures key metrics such as download speed. The IPD is a repeated game that allows the players to achieve mutual gains from cooperation, but it also allows for the possibility that one player will exploit the other, or the possibility that neither will cooperate[3].

There has been work on analytical models of BitTorrent with heterogeneous bandwidth classes [8], [19]. However, these do not model the BitTorrent protocol as a strategy in a game. Thus, it is not clear whether peers employing Tit-For-Tat[3], to play the Prisoner's Dilemma is the right model for BitTorrent. 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. Hence it has been proposed that BitTorrent might be better modeled as an auction [18]. With heterogeneous classes of peers in the system, it is not clear whether the interaction between a high class peer and a low class peer should be modeled as a PD, or perhaps, as the Dictator game, a game in which only one player has any strategic input into the outcome. Do peers in BitTorrent generally apply the TFT strategy or is it more aptly described as Out-For-Tat (OFT)?[1] Thus, we seek to address the following points: Can we convincingly argue that considering BitTorrent as a strategy in iterated games, helps us gain insights from the vast amount of literature on such games, from diverse fields like psychology, sociology and economics? Can we develop and validate an analytical model that considers BitTorrent as a TFT like strategy? Can a strategy model of BitTorrent be developed that explains aspects of BitTorrent hitherto deemed contradictory to TFT? Can such a model be used to provide new insights into BitTorrent's incentives and lead to protocol improvement?

We present a model of BitTorrent that allows us to gain new insights into the protocol. Specifically, given our understanding of the TFT policy in BitTorrent, we demonstrate that contrary to previous work [10], TFT as implemented in BitTorrent is not a Nash Equilibrium. We also come up with a variant that we call Birds, which is a Nash Equilibrium.

[1]OFT is a strategy similar to TFT. However, it differs from TFT in the respect that instead of retaliating to a partner's defection, it exits the game to look for another partner

To Nash or not to Nash?

We would like to understand: Given the complexity of modeling a single protocol using game theory, given the amount of uncertainty prevailing over a single protocol's incentives in the P2P literature, how feasible is it to suppose that traditional game theoretic analysis can be easily applied for modeling and analyzing a variety of P2P protocols? How feasible is it to use an equilibrium analysis that, for tractability purposes, often relies on single points in the design space for analyzing the robustness of protocols against exploitative variants?[2]. How realistic is it to suppose that such an analysis, done on one protocol might provide insights for the design of other protocols? How reasonable is it for such an analysis to assume non-rational players; those that are not necessarily maximizing their rewards, but whose strategies "may simply reflect standard operating procedures, rules of thumb, instincts, habits, or imitation" [3]. Furthermore, given a set of equilibria, can a static analysis indicate which equilibria are more likely to be selected over some others; specifically, can the dynamics of strategies over time be predicted and can learning and behavior change, over time, be included as part of such an analysis? On all the above counts, we argue that game theoretic analysis fares badly.

The PRA approach

We propose an alternate approach which rests on two features: 1) A generalized multi dimensional strategy space that can characterize when, with whom, how many and how much, peers can cooperate and which can be applied to a large variety of P2P Systems, and 2) A simulation based framework for analyzing protocols developed over this space. In our view, this two-pronged approach gives designers a general tool for helping them design new protocols; for testing their performance; and for analyzing their robustness against potential variants. Indeed we aim to explore if we can achieve the goal envisaged in early pioneering P2P work[11] for creating a "common framework, including the formalization of adversarial models, definitions of robustness etc" for analyzing and evaluating incentive schemes, using a practical approach.

In order to achieve this goal, taking inspiration, not just from traditional game theory but also from the works done on cooperation in Biology, Psychology and Social Simulation, we construct abstractions for the most important features and parameters of reciprocation strategies that could be employed in P2P systems. We argue that our design space is general enough to be applied to a variety of P2P Systems.

Furthermore, by using a simulation based methodological approach for analyzing this space, we provide a framework for designers to design and analyze new protocols. We call it the *Performance, Robustness, Aggressiveness* (PRA) analysis. Here, we note that there has been early work on analyzing the design space

[2]So for example in the specific case of BitTorrent, a broad characterization for rate assignment strategies has been studied [10]. However, its not quite clear how the change of other protocol parameters could have an impact on such an analysis.

for P2P networks using evolutionary techniques[12], it was limited to evaluating simple games on a subset of a limited design space. Our analysis occurs at a deeper level of protocol design and involves several dimensions that are likely to be present in implemented protocols. Also, in our view our approach is more reasonable than an evolutionary approach, in that it analyzes performance and robustness more comprehensively.

2.2 A Space of P2P Protocols

We wish to design a distributed protocol, which maximizes performance of the system under the assumption that protocol variants may enter the system. This challenge can be compared to Distributed Algorithmic Mechanism Design. However, rather than propose a single protocol we define a design space over a set of salient dimensions, which affect the incentive structure. We therefore focus on those aspects of the protocol, which require cooperative peer interaction. The outcome of these interactions determines the performance of the system. We have identified the following salient dimensions applicable to a large variety of P2P systems. We assume that peers have access to private and/or shared storage.

Peer Discovery: In order to perform productive peer interactions, it is necessary to find others, 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. In BT a peer locates other peers by contacting a central server called the Tracker. Peers may also discover each other using a distributed mechanism called PEX (Peer Exchange) and / or a DHT (Distributed Hash Table).

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. In BT an "optimistic unchoke" policy serves as a stranger policy. Random strangers are selected every 3 rounds (normally 30 seconds). The number of peers to optimistically unchoke can be fixed or a function of bandwidth. Therefore, BT follows an always cooperate strategy against strangers.

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. BT implements the selection function using the regular unchoke policy. The regular unchoke policy uses a *candidate list*, *a ranking over this list* and a value for the number of peers to select. The candidate list includes all those peers who provided data to the peer in the last round. The ranking is in order of the fastest first. The number of partners is usually set to 4 or a function of the peer's upload bandwidth.

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. In

BT a peer divides its resources (upload bandwidth) equally over all peers that it cooperates with in a given round of interaction.

2.3 The PRA Analysis Approach

We can characterize any protocol, from a given design space, over three measures (or dimensions). For a given protocol Q, the three measures we choose are:

- Performance - overall performance of the system when all peers execute Q
- Robustness - ability of a majority population executing Q to outperform a minority subpopulation executing a protocol other than Q
- Aggressiveness - ability of a minority population executing Q to outperform a majority population executing a protocol other than Q

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 Q can be characterised as a point within a three dimensional Performance, Robustness, Aggressiveness (PRA) space.

It is desirable, in open systems, to design protocols which maximise all three measures. However, it is clear 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. For example, an early BitTorrent variant was supposedly more robust but was said to lower performance. Similarly, an effort based variant of BT achieves higher performance than normal BT but is seemingly not robust.

We now define more precisely how we can map a given protocol Q, which can be expressed as a point in the design space, to a point in the PRA space. We assume that for each peer in a system of peers we can calculate a utility which quantifies individual performance.

Given this we define the performance (P) of protocol Q as the sum of all individual utilities in a population of peers executing Q normalised over the entire protocol design space. Hence where P = 1 this indicates the best performance obtained from any protocol in the design space.

We define the Robustness (R) for protocol Q as the proportion of all other protocols from the design space that do not outperform Q in a "tournament". A tournament consists of a mixed population of peers executing one of two protocols (where Q is in the majority). The winning protocol is that which obtains the higher average utility for the peers executing it.

Aggressiveness (A) is similarly defined as Robustness but here Q is in the minority.

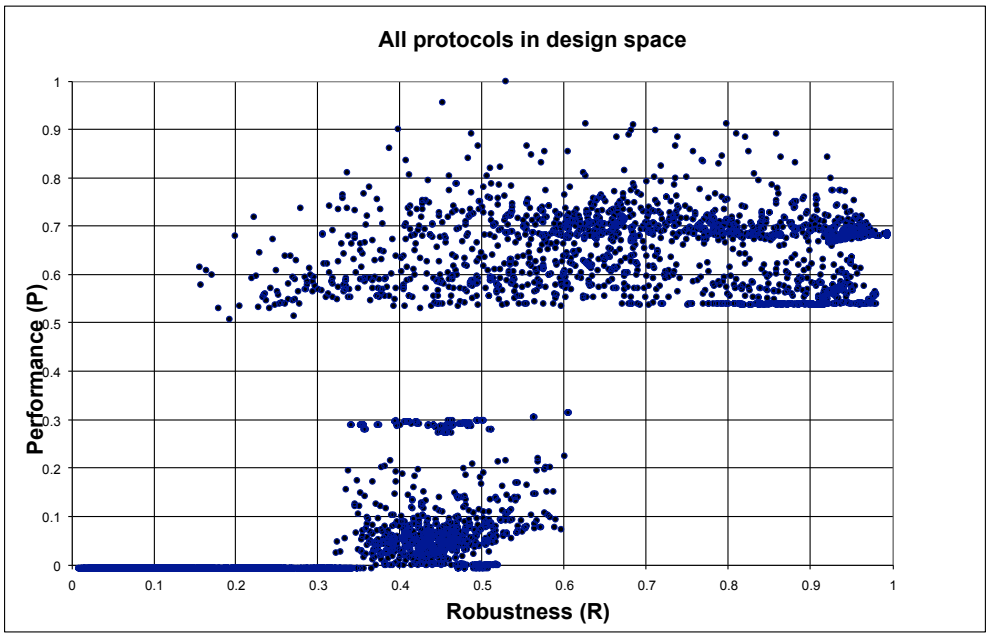


Figure 2.1: Scatter plot of all approx. 4000 protocols in the design space with robustness (R) against performance (P). Note the two clusters indicating two classes of protocol (one with low performance and medium robustness and one with higher performance and a broader range of robustness including highly robust). The underlaying mechanisms within the two classes are currently under investigation. The performance and robustness measures are defined in the text. The results presented here are a synthesis of over 160 million individual simulation runs.

2.4 BitTorrent Protocol Space - Some Initial Results

We defined a space of BT protocols over the dimensions previously discussed. We do not give details of these dimensions here (this will be discussed in detail in the forthcoming technical paper) however, they are sufficient to capture many BT protocol variants including several already proposed and deployed variants and a large space of novel variants which have not been examined before. Overall the space comprised approx. 4000 protocol variants. Simulation runs were performed to determine performance (P) and robustness (R) measures using realistic bandwidth distributions and multiple runs (ten) to reduce variance due to noise. For the robustness experiments this required approx. 160 million individual runs. A scatter plot of results can be seen in figure 2.1.

2.5 Summary

In this chapter we have briefly summarised on-going work that applies an approach, which we consider quite general, for exploring a space of protocol vari-

ants. We hope to have given a flavour for the results that are being obtained. Detailed descriptions and analysis 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 [14].

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 [20] has shown that most BitTorrent commu-

nities 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 [2], [15], 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.* [4].

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.* [24] and Meulpolder *et al.* [19]. The notation we use is shown in Table 3.1. Similar to the approach in [19], 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:

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

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:

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

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 $\mu_1 < \mu_2 < ... < \mu_N$ and $D_1 < D_2 < ... < D_N$.

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

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

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.:

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

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

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

while for seeders who adopt the RS policy it holds:

$$\beta_{ij} = u_i \pi_j. \tag{3.7}$$

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:

$$\begin{aligned}n_i &= \frac{\sum_{1 \le j \le N} \alpha_{ji} x_j + \beta_{ji} y_j}{x_i}, \\ d_i &= \frac{D_i}{n_i}.\end{aligned} \tag{3.8}$$

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

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

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

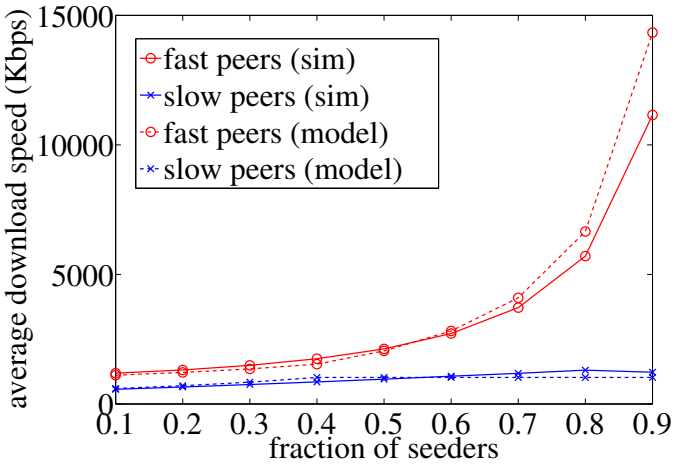


Figure 3.1: The average download speeds of fast and slow peers in a system with 50 fast peers and 50 slow peers, for different fraction of seeders. The capacities of peers are the following: 1024 Kbps up and ∞ down for fast peers; 512 Kbps up and 1024 Kbps down for slow peers. Seeders use the FF policy.

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 allocate 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 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:*

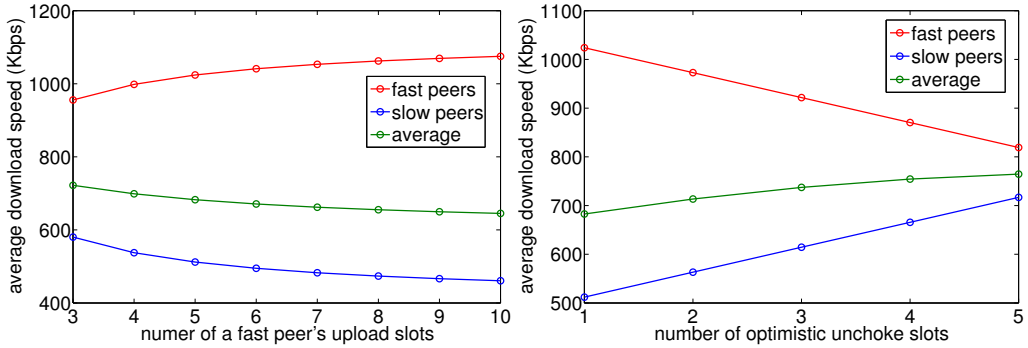
$$\frac{\mu_f u_s}{\mu_s u_f} = \frac{u_f^{(op)} + u_s^{(op)}}{u_f^{(op)}}. \tag{3.10}$$

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.:

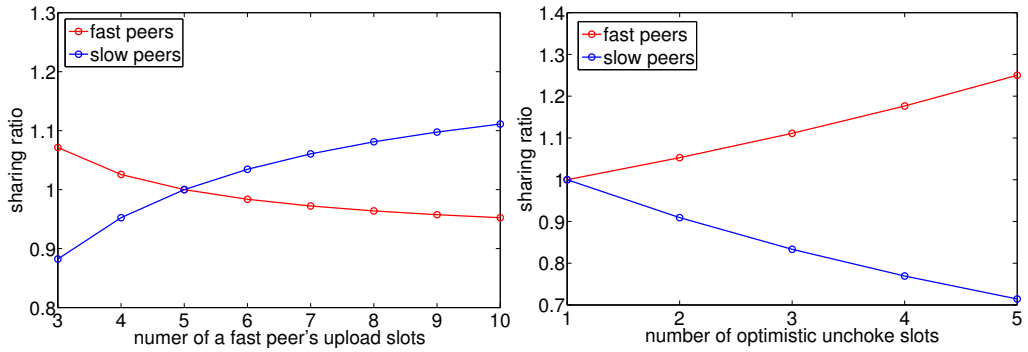
$$\frac{\mu_i}{\lambda_i F/x_i} = \frac{\mu_i x_i}{F \sum_{j\in\{f,s\}} D_{ji} x_j \mu_j}. \tag{3.11}$$

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

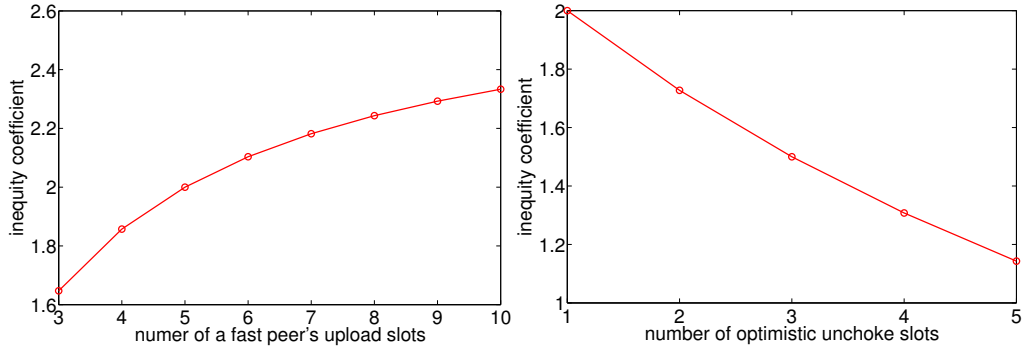
$$\frac{\mu_f x_f}{\sum_{j\in\{f,s\}} D_{jf} x_j \mu_j} = \frac{\mu_s x_s}{\sum_{j\in\{f,s\}} D_{js} x_j \mu_j}. \tag{3.12}$$



(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.

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.

□

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

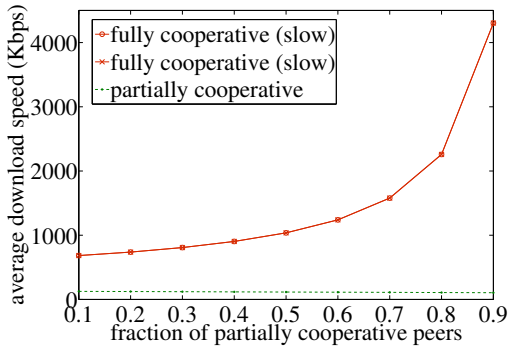
Here we analyze the influence of having all peers open more optimistic unchoke slots. While peers always open 5 unchoke slots in total, we let the number of their optimistic unchoke slots vary from 1 to 5. As we can see in Fig. 3.2(a), in this way the download speed of slow leechers is improved by 40%, 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.

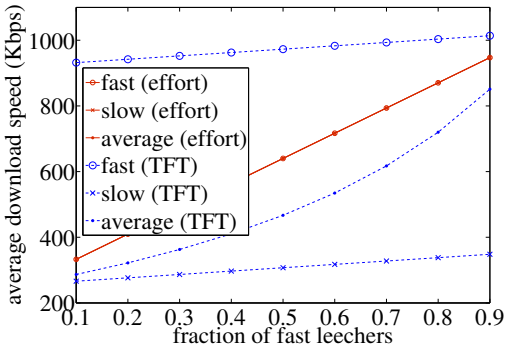
Strategy 3: reducing inequity by replacing TFT with effort-based incentives

Rahman *et al.* [25] 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.

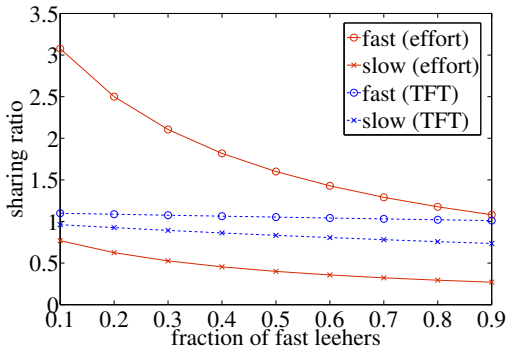
Consider that there are two types of peers in the system, peers that contribute all their upload bandwidth (*fully cooperative*) and peers that only contribute a fraction of it (*partially cooperative*). Let n_p represent the number of partially cooperative peers, and n_{ff}, n_{fs} represent the number 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 unchoking them. The slot allocation for each class of peers can be calculated as:



(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.

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

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)).

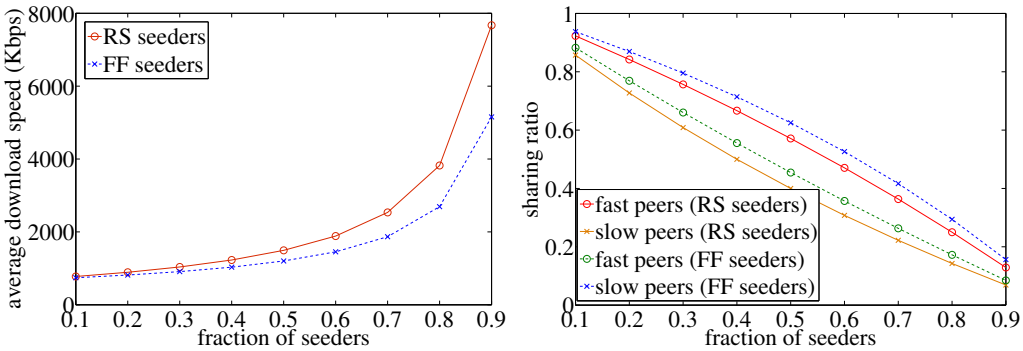
Strategy 4: enhancing reciprocity or reducing equity with a seeder's policies

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 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)).

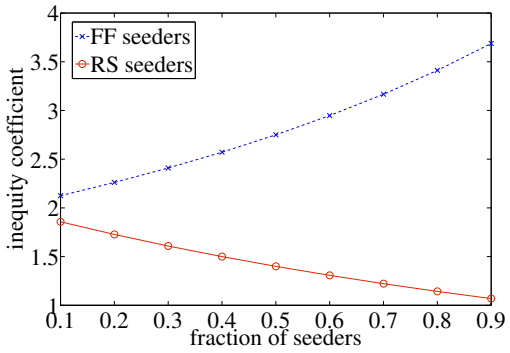
3.4 Related Work

There are a number of studies on modeling and improving BitTorrent's incentive policies. Some earlier work focuses only on homogeneous systems [17], [33], [24]. In [30], the authors consider heterogeneous BitTorrent systems, but only with two classes of peers. Fan *et al.* [4] have developed a general heterogeneous model to evaluate the tradeoff between performance and fairness. Meulpolder *et*



(a) Average download speed

(b) Sharing ratio



(c) Inequity

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

al. [19] and Chow *et al.* [8] 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 [14]. 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] http://en.wikipedia.org/wiki/nashequilibrium.

[2] A. Vlavianos, M. Iliofotou and M. Faloutsos. BiToS: Enhancing BitTorrent for Supporting Streaming Applications. In *Proceeding of IEEE Global Internet Symposium*, 2006.

[3] R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.

[4] 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.

[5] M. Barthélemy. Betweenness centrality in large complex networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):163–168, 2004.

[6] S. Buchegger and J. Le Boudec. A robust reputation system for mobile ad-hoc networks. *Proceedings of P2PEcon, June*, 2004.

[7] 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.

[8] A. Chow, L. Golubchik, and V. Misra. Bittorrent: an extensible heterogeneous model. In *INFOCOM 2009, IEEE*, pages 585–593. IEEE, 2009.

[9] 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.

[10] B. Fan, D. Chiu, and J. Lui. The delicate tradeoffs in bittorrent-like file sharing protocol design. In *Network Protocols, 2006. ICNP'06. Proceedings of the 2006 14th IEEE International Conference on*, pages 239–248. IEEE, 2007.

[11] M. Feldman and J. Chuang. Overcoming free-riding behavior in peer-to-peer systems. volume 5, pages 41–50. ACM, 2005.

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

[13] L. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] M. Meulpolder, J. Pouwelse, D. Epema, and H. Sips. BarterCast: A practical approach to prevent lazy freeriding in P2P networks. 2009.

[22] 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.

[23] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, 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.

[24] D. Qiu and R. Srikant. Modeling and performance analysis of bit torrent-like peer-to-peer networks. In *ACM SIGCOMM*, 2004.

[25] 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.

[26] 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.

[27] 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.

[28] J. Shneidman and D. Parkes. Rationality and self-interest in peer to peer networks. pages 139–148. Springer, 2003.

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

[30] 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.

[31] Wikipedia. Ford-fulkerson maxflow algorithm. http://en.wikipedia.org/wiki/Ford-Fulkerson_algorithm.

[32] 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.

[33] Y. Tian, D. Wu and K. W. Ng. Modeling, analysis and improvement for bittorrent-like file sharing networks. In *Proceedings of INFOCOM*, 2006.