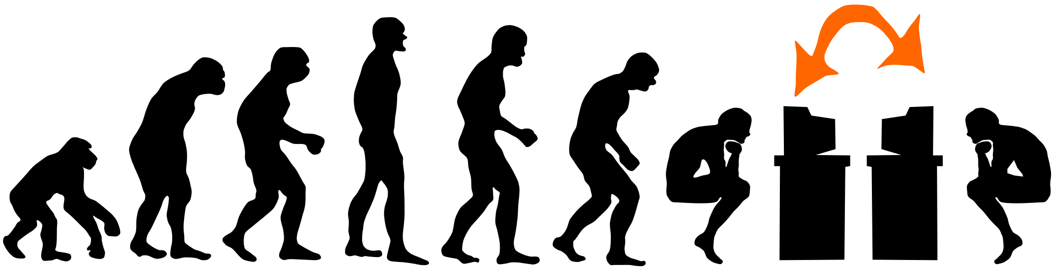
Incentivizing Seeding In BitTorrent

Indirect Interaction as an Incentive to Seed

In Search for Homo Swappus



Arman Noroozian

Incentivizing Seeding In BitTorrent

Thesis submitted in partial fulfillment of

the requirements for the degree of

Master of Science in Computer Science

by

Arman Noroozian

31 August 2010

TUDelft
Algorithmics Research Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Incentivizing Seeding In BitTorrent

Author:	Arman Noroozian
Student id:	1334654
Email:	arman.noroozian@gmail.com

Abstract

BitTorrent has turned into the most popular P2P file sharing protocol and is used for various purposes such as *Video on Demand* and *Media Streaming*. The fundamental problem with P2P networks in general is that quality of service highly depends on altruistic resource sharing by participating peers. Many peers freeride on the good intentions of others and BitTorrent is no exception. Current solutions like reputation systems and sharing ratio enforcement are complex, exploitable, inaccurate or unfair at times. The need to design scalable mechanisms that mitigate such problems is evident. We demonstrate through measurements that BitTorrent peers are able to barter pieces of different files (indirect interaction) which is a previously unknown property of the BitTorrent protocol. We introduce a centralized extension for the BitTorrent protocol which we refer to as the indirect interaction mechanism (IIM). IIM incentivizes seeding and mitigates problems of unfairness and exploitation while at the same time achieving linear scalability. We provide game theoretical models of the mechanism and demonstrate through analysis and simulations that IIM improves BitTorrent performance for certain cases and that it does not degrade performance for others. We conclude that IIM is a practical solution to the fundamental problem of P2P networks like BitTorrent.

Thesis Committee:

Chair:	Prof. Dr. Cees Witteveen, Faculty EEMCS, TU Delft
University supervisor:	Dr. Mathijs de Weert, Faculty EEMCS, TU Delft
	Dr. Yingqian Zhang, Faculty EEMCS, TU Delft
Committee Member:	Dr.ir. J.A. Pouwelse, Faculty EEMCS, TU Delft

Acknowledgements

I would like to thank my family and Ms. Saman Sattari Khavas for supporting me during my studies. A special thanks again goes to Saman and to Mr. Hadi Asghari for their friendly insights and all other friends that have made my stay in Delft a memorable experience.

I would like to thank my supervisors Prof. Dr. Cees Witteveen, Dr. Mathijs de Weert, Dr. Yingqian Zhang and Dr. Johan Pouwelse for their guidance, daily supervision and patience. In addition I would like to thank Michel Meulpolder for permission to use his BitTorrent simulator and Dr. David Hales and Boxun Zhang for providing insights into some of the subjects of this research. Finally I would thank the Tribler team for providing some of the material that has been used in this research.

Arman Noroozian
Delft, the Netherlands
31 August 2010

Contents

List of Figures

Chapter 1

Introduction

The demands from the Internet have grown extremely high in quantity and there is a wide range of service and quality expectations. Internet usage varies from browsing, entertainment, Internet TV and P2P file-sharing to e-business, scientific research, collaboration, communication, media streaming, telephony, etc. These services are made accessible from a wide range of devices with various connection speeds which generate a large volume of traffic on the Infrastructure. Studies of the network traffic actually show that more than 40 percent of the traffic on the Internet is originated from P2P File-Sharing software [3]. BitTorrent alone, which is one of the most popular P2P file-sharing applications, is roughly responsible for a third of the total Internet traffic [3, 2]. We have chosen BitTorrent as the subject of this research because of it popularity and wide usage.

The P2P paradigm has reshaped Internet services from a client-server based architecture to a generalized Peer-to-Peer architecture. In P2P networks, pairs of peers play the role of clients and servers for each other during a series of transactions. That is, peers can be viewed as possessing a collection of attractive resources which they serve to other peers and in return seek the attractive resources of others. Unique identifiers are assigned to peers to assist with their location (usually a combination of their IP address and port). By sharing resources (bandwidth in the case of BitTorrent), peers reduce the burden of a single centralized server in meeting the demands of all clients and hence improve scalability. P2P networks are able to meet the demands of a much larger number of peers in comparison to traditional client-server based architectures.

1.1 Problem Statement

A shift to P2P technologies immediately brings an important problem to the foreground. The problem with P2P networks is that quality of service highly depends on altruistic resource sharing by participating peers. That is, such networks heavily rely on peers to voluntarily give away their resources for the common good and BitTorrent is no exception. It has been shown that many peers exploit this fact and freeride on the good intentions of others [20, 4, 31] without contributing any resources themselves. The ability of peers to change identities and hide their malicious behavior (whitewashing [20]) in addition to freeriding and other types of malicious behavior constitute some of the fundamental problems of P2P networks and re-enforce the need to limit the impact of such behavior.

BitTorrent in particular has been able to address some of the fundamental problems of P2P networks by taking a different approach to file-sharing. Unlike previous generations of P2P file-sharing application BitTorrent peers exchange pieces of a single file for pieces of the same file within the framework of a tit-for-tat (TFT) mechanism. BitTorrent Peers can still download files in parallel nevertheless they only exchange pieces of a file for pieces of the same file. Even though the TFT mechanism has been able to mitigate the freeriding problem [10, 29] researchers have still outlined possible ways of freeriding in BitTorrent and have demonstrated that it does exist [23, 31, 50, 53, 30].

Such fundamental problems of P2P networks have lead to the development of various types of reputation systems and monetary mechanisms that limit malicious behavior and incentivize seeding. General purpose reputation systems such as EigenTrust [26] and BarterCast [34] have been designed with special care to maintain scalability. In addition, monetary mechanisms have been developed that in combination with reputation systems incentivize honest reputation feedback [25, 51] but have not gained popularity due to the need for complex accounting and a central banking entity.

The main problem with reputation systems and monetary mechanism is that they are exposed to vulnerabilities ranging from simple lying to more complex collusion [25], identity theft and Sybil [12] attacks. In addition reputation systems can be used as a platform to launch DDoS attacks by falsely assigning an exceptionally high reputation score to the target of the attack which may not necessarily be part of the P2P network. Moreover, most reputation systems are at best partially accurate and require numerous messages to be passed between the peers and other entities in the network. This damages the scalability of the network as a whole. Finally it has been shown that any global history reputation system that computes reputation scores non subjectively cannot be sybil-proof [8].

Clearly, a drive for better quality of service exists among BitTorrent users as more and more users are attracted to private BitTorrent communities (i.e. TVTorrents, FileList, etc.) in which membership is by invitation and requires a high sharing ratio to maintain membership [33]. However such communities suffer from unfairness and discrimination against slower peers by faster peers (BitCrunch [24]) as a minority of faster peers can result in long periods of waiting for slower peers to gain access to resources even if slower peers are fully compliant with the protocol (non malicious).

To this date we are unaware of any mechanism specifically designed for BitTorrent that incentivizes seeding in a more public domain and solves the vulnerabilities and fairness issues that were outlined earlier . The exception to this statement is Tribler [1] which is a highly specialized BitTorrent client that implements a sense of community among its users and implements the BarterCast reputation system to incentivize seeding. While Tribler is slowly gaining popularity among users the need for mechanisms that operate in a public domain and incentivize seeding beyond the TFT mechanism of BitTorrent is evident. Such mechanisms have to be scalable and in addition to address the vulnerabilities that emerge in reputation systems such as BarterCast and EigenTrust.

1.2 Research Goals

As a result of what we have outlined in the previous section we define our main research goal to be as follows:

> *Design a mechanism for incentivizing seeding in BitTorrent that does not degrade the Standard BitTorrent protocol's performance, is fair, operates in the public domain and does not suffer from known vulnerabilities. The mechanism should be scalable and/or easily extensible to a fully distributed setting.*

Our research goal can be divided in to the following sub-questions that naturally relate to our research goal. We will use these questions as criteria that help us evaluate how well we have achieved our goal:

- *How is seeding incentivized?*: Seeding should be incentivized by providing advantages to peers that choose to seed in comparison to peers that do not do so. The advantages should be in terms of a better quality of service (e.g. faster download times). Naturally malicious peers should not be given advantages.

- *What is an acceptable performance for the mechanism?*: The mechanism should not significantly degrade the current performance of BitTorrent in terms of the time required to download a file. By significance we mean significance in term of statistical comparison of mean performances (See Chapter 6 for details).

- *What is fair?*: In addition to mitigating freeriding the mechanism should not allow more resourced peers to discriminate against less resourced peers that follow protocol.

- *What is the public domain?*: The mechanism should be implementable on public trackers and private trackers alike without the need for unique identifiers that are used in private trackers.

- *What are the known vulnerabilities?*: The BitTorrent protocol and the earlier outlined mechanisms suffer from vulnerabilities such as the Large view exploit[50], Sybil attacks, DDoS Attacks and possibility of Collusion. Except for collusion attacks which are much harder to implement in practice, the mechanism should be immune to the other types of attacks. We will not consider collusion attacks in this research.

- *What is scalable?*: A constant scaling factor is the ultimate goal for the mechanism. That is the performance of the mechanism remains constant as the number of peers grow. However a linearly decreasing performance (linear scaling) is also acceptable and can be considered a partial fulfillment of the scalability requirement. In case non of the previous requirements is achieved we consider the extensibility of the mechanism to a fully distributed setting with the help of existing technologies a partial fulfillment of the scalability requirement.

1.3 Methodology and Contributions

To achieve our research goal we have firstly conducted a measurement study of the FileList.org private BitTorrent tracker. The motivation for this measurement being previous studies that indicate that many BitTorrent peers participate in multiple swarms (multi-swarm) simultaneously [22]. The aim is to demonstrate that BitTorrent peers are essentially able to indirectly interact. Indirect interaction is the ability to barter pieces of multiple files in contrast to exchanging pieces of the same file which is how BitTorrent operates currently. Based on what we have found in this study we have designed a centralized mechanism for incentivizing seeding in BitTorrent which we refer to as the *indirect interaction mechanism* (IIM). We have based our mechanism on the principles of complexity in distributed algorithmic mechanism design (DAMD) as outlined and largely agreed upon in [11, 16, 17, 37, 41] (See chapter 2 for details). We have modeled our mechanism in a game theoretical setting and studied the model through analytical, experimental and statistical methods to evaluate the properties of the mechanism.

Our main contributions in this research can be stated as follows:

- A new measurement study on the properties of the BitTorrent protocol. We demonstrate previously unknown properties of the protocol in this measurement study and show that indirect interaction is possible with high probability.
- A centralized mechanism for incentivizing seeding in BitTorrent (IIM) that can be extended to a fully distributed setting with existing technologies such as DHTs and BuddyCast [44].
- A new game theoretical model of the BitTorrent protocol. More specifically we present a probabilistic model of BitTorrent and demonstrate that the model is able to explain many of the well known and observed phenomena in the BitTorrent protocol.
- A game theoretical model of the indirect interaction mechanism (IIM). This model is an extension of the probabilistic model for the BitTorrent protocol. We analyze this model and hypothesize about the properties of IIM.
- An experimental economics model of IIM. We use this model to study human behavior with respect to IIM.

1.4 Results

Our measurement study of the FileList.org tracker demonstrates that BitTorrent peers are essentially able to indirectly interact. Indirect interaction is a direct consequence of peers being active in multiple file sharing swarms (multi-swarming) as BitTorrent peers barter in swarms. Only indirect interactions that occur in a closed loop (cycle) are of interest in our measurement study. We measure the probability of indirect interaction for cycles of various lengths and demonstrate that there is a high probability that such interaction can occur. We generalize the results of the measurement study to public trackers because of the soft sharing ratio enforcement policies of the FileList.org tracker in addition to other studies that demonstrate a high percentage of multi-swarming also occurring in public trackers [22]. Consequently, our measurement study forms the basis for the design of our indirect interaction mechanism (IIM).

Analysis of our game theoretical model of IIM leads to two hypotheses indicating that peers are better off by interacting indirectly when multi-swarming. We hypothesize that for certain cases indirect interaction leads to Bayesian Nash equilibria in a homogeneous setting (when all peers have the same upload/download bandwidth). Additionally we hypothesize that indirect interaction will not be significantly outperformed by BitTorrent for other cases in the homogeneous setting. We verify our former hypotheses by simulating the indirect interaction mechanism in a homogeneous setting and argue why BitTorrent should not outperform IIM for other cases. Moreover, we simulate IIM in a heterogeneous setting to study the properties of the mechanism and demonstrate that as long as the bandwidths of the indirectly interacting peers are the same, benefits similar to the homogeneous setting exist. We also find out that as long as the difference in bandwidth for indirectly interacting peers is not too large IIM will perform no worse than standard BitTorrent and will not be significantly outperformed. These results lead us to conclude that IIM incentivizes seeding in BitTorrent.

In order to predict actual user behavior (as opposed to rational agents in the game theoretic sense) we also consider modeling IIM as an experimental economics experiment in which humans are involved. We partially succeed in demonstrating (through previously conducted experiments) that humans should react positively towards IIM and choose to indirectly interact instead of disconnecting from the bartering swarms.

1.5 Outline

The remainder of this document is organized as follows. We first provide some background information on BitTorrent, reputation systems, game theory and some principles of mechanism design in Chapter 2. In Chapter 3 we present our measurement study of the FileList.org tracker. Chapter 4 provides the detailed design of our indirect interaction mechanism and presents methods by which the mechanism could be extended to a fully distributed setting. Chapter 5 is dedicated to our game theoretical and experimental economics model of IIM. We provide our analysis of the models and main hypotheses in this chapter. Chapter 6 provides the results of our simulations of IIM in a homogeneous and heterogeneous setting. We verify our hypotheses in this chapter. Finally we provide a summary of the results, our concluding remarks and recommendations for future work in Chapter 7.

Chapter 2

Background

This chapter is dedicated to providing some of the background knowledge that is required to understand the topic of this thesis dissertation and problems that it solves. Most of the terminology and concepts that we use in our research is summarized in this chapter. We present basic concepts of the BitTorrent protocol in Section 2.1 which are regularly used throughout this text. We present basic knowledge of reputation systems, private BitTorrent communities and their vulnerabilities in Section 2.2. Furthermore, we introduce some notions from game theory in Section 2.3 which are used to analyze our models and are essential to understanding how the research solves the identified problems. Finally in Section 2.4 we present some important principles of mechanism design which are important to understanding issues of complexity and scalability.[1]

2.1 Introduction to BitTorrent

BitTorrent is a File-Sharing Protocol designed by Bram Cohen [10] that uses a tit-for-tat (TFT) mechanism to distribute content between peers. In simple terms, the TFT mechanism is to provide the same service to peers as the service received from them. The choice of the TFT mechanism has been a main success factor in reducing freeriding in P2P networks [29, 10].

BitTorrent is made up of 3 basic entities: *trackers*, *seeders* and *leechers*. Trackers are the central authority of the protocol and are responsible for coordinating file-sharing among peers. Additionally trackers keep track of the peers that have joined the swarm and basically gather data from each peer's activities. Peers are divided into two types: seeders and leechers. Seeders are a special type of peer that possess an entire file and are distributing the file for free. Leechers are peers that have obtained parts of a file and are in the process of acquiring the rest of the parts. Leechers do not give away parts of the file for free and expect parts in return (TFT). This being said a file is split to smaller pieces (also referred to as chunks) that are exchanged for other pieces. BitTorrent terminology also includes the concept of *swarms*, which is a collection of peers (seeders and leechers) that have formed

[1] Please note that this chapter is a summary of an extensive literature survey submitted before this thesis dissertation.

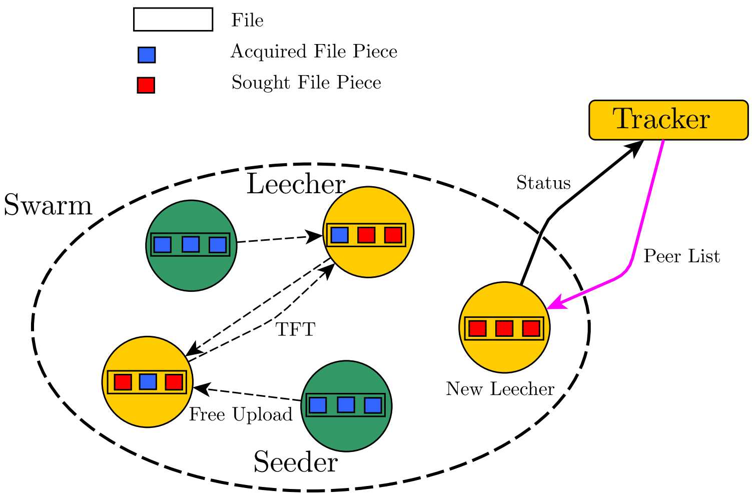


Figure 2.1: The BitTorrent Protocol [1].

around the task of distributing a *single* file (see Figure 2.1). Trackers can keep track of several swarms.

To distribute a file over BitTorrent some initial configurations are required. Firstly, a tracker must be created or a previously existing tracker must be considered.[2] Secondly, a *.torrent* file must be created. The *.torrent* file contains information about the network address of the designated tracker and information about the file that is to be distributed *e.g. file size, number of pieces* and *hash values for each piece, etc.* At this stage the first seeder must be added to the tracker's list of peers by the creator of the *.torrent* file. Users who want to download the file have to find and provide the *.torrent* to their BitTorrent clients to start downloading. The *.torrent* files can be found through various methods e.g. search engines, sharing communities, portals, etc.

Once a BitTorrent client has received the required information for download (the .torrent file), it will contact the designated tracker to get a list of peers that can assist with downloading the file. According to the protocol a peer may contact the tracker approximately every 30 minutes to acquire a new list of peers. The list of peers is generated every time by randomly choosing a constant size subset of peers from the set of all available peers in the swarm. The list typically contains 50 peers. Once a client has a list of peers which it can contact, it will try to establish a connection with them (its neighbors) and acquire a piece of the file from anyone of them. Seeders are willing to give a piece of the file to the newcomer for free. This process is referred to as *bootstrapping*. A second possible way of bootstrapping also exists which we explain further on. Once the newcomer has bootstrapped, it can start bartering piece(s) for other pieces. Simultaneously it can continue to get pieces for free from seeders as well. Peers maintain a list of the pieces that they currently possess and advertise this to other peers to inform them of available pieces and pieces that they still seek to download (see Figure 2.1). Peers also inform the tracker about their status, i.e. how much

[2] The *OpenBitTorrent* tracker is an example of a free tracker available for public use

of the file they have downloaded and how much they have uploaded.

Every peer has a limited number of connections which are *choked* by default (i.e. no pieces are bartered on choked connections). The available upload bandwidth of a peer is split into equal *slots* which are used to upload pieces on some of the connections (unchoked connections). As a result peers must utilize their unchoke slots. A leecher in particular will upload pieces to neighbors that have also been uploading pieces to it. As explained before this constitutes a tit-for-tat mechanism which is implemented in the *unchoking algorithm*. The unchoking algorithm servers a two-fold purpose. First, the unchoking algorithm presumes that there are a number of available upload slots and determines the neighbors who can be unchoked. To determine which neighbors get the slots, the process is presumed to follow rounds of a constant length in seconds (typically 15 seconds). Every peer keeps track of the amount of bandwidth it has received from its neighbor in each round. In return for this bandwidth a peer will give an upload slot to requesting neighbors who provided more bandwidth to it in the previous round.

The unchoking algorithm's second purpose is referred to as *Optimistic Unchoking*. Every 3 rounds the unchoking algorithm puts aside one of the available upload slots and always assigns this slot to some random neighbor, regardless of the amount of bandwidth it received from that neighbor in the previous round. This allows a peer to explore other peers in the network and find ones that are more cooperative or have higher upload bandwidth. Optimistic unchoking also helps with bootstrapping a new coming peer.

More details about the BitTorrent protocol can be found in Section 4.1 and ultimately in the BitTorrent protocol specification [9]. While BitTorrent is one of the first P2P protocols that has successfully reduced freeriding behavior it is in many cases accompanied by additional mechanisms that help improve the protocol. In our next section we survey some of the most important ones that relate to our research and problem description.

2.2 Related Work

BitTorrent's TFT mechanism is a successful method of incentivizing contributions by leechers during a download process. However, this success does not extend beyond the boundaries of a single swarm. That is, seeders that already possess and entire file do not require their contributions to be reciprocated in the framework of the TFT mechanism. Measurements show that many leechers disconnect from the network shortly after they have downloaded the entire file [43, 22, 53]. Since the seeding of content is of major importance in the performance of BitTorrent a dependency on altruistic behavior is a major problem in BitTorrent and P2P networks in general [19, 18]. As a result additional technologies sometimes accompany the protocol that extend the incentives to include benefits for seeding content. One of the most popular scientific methods of achieving this goal is the use of reputation systems[3] in which seeders gain reputation scores that will benefit them in their future interactions (i.e. when leeching new content).

[3] Except for Tribler we are unaware of other BitTorrent clients that implement reputation systems. Reputation systems are less popular in practice to this date.

2.2.1 Reputation Systems

Well designed reputation systems could severely mitigate misbehavior while demanding minimal cost from the well-behaved users [32]. Indeed, a great deal of effort has been put into designing reputation systems for P2P systems that impose minimal additional complexity. Reputation systems are incorporated into P2P systems to degrade service to unpopular peers (*e.g.* freeriders), hence providing the necessary incentive to seed content. Unpopular peers are distinguished from reputable peers by keeping a history of each peer's interactions. However, if the history record is too short, peers can build up reputation to exploit the system at a later stage [32, 19]. On the other hand if the history record is too long, peers can misreport or drop reports of well behavior to degrade the service of obedient peers. Furthermore, decisions must be made that define the policy towards peers that have no reputation yet [32]. In what follows we give some background information on reputation systems and different approaches to designing them. We give background information on the EigenTrust [26] and BarterCast [34] as examples of reputation systems and we identify problems with using reputation systems in general.

Components of a Reputation System

Any well designed reputation system needs to have at least three components: *information gathering*, *scoring/ranking* and *response* components [32]. The usual assumption is that information is gathered by keeping track of peer transactions.

Obviously information gathering through tracking transactions between peers requires them to have some sort of identity. Therefore, identity systems can be considered as a subcomponent. Peers require sufficiently persistent identities to be able to tie actions to behavior in the history records. However, anonymity, that is, a disconnection between the actual identity of a peer in the real world and identity in the system is also required. Furthermore, Peers should not be able to impersonate other peers by spoofing and finally identities should not be forgeable. Some of these requirements are obviously in conflict with the goal of the reputation system. For example total anonymity cannot be achieved in a feasible reputation system [32].

The information gathering component should further deal with information sources. In general the quantity and quality of information from sources are diametrically opposed [32]. That is, as more information is gathered from the sources the credibility of each piece of information lowers. Therefore some reputation systems adopt a local history where each piece of information is more credible and others adopt a global history where information is less reliable. Moreover, as it is impossible to enforce honest, accurate reporting on transaction outcomes by all peers [32] the information gathering component should also deal with the integrity of the data. Finally depending on the approach, information gathering should also include some policy towards strangers and how to deal with them. When no information is available about the reputation of a specific peer this policy helps the other party to make a decision.

Regarding the scoring and ranking component, there are issues of representation; i.e. what input and output factors should be considered. One can state that the primary purpose of the reputation score is to help a peer decide which other available peer it should interact with. In an ideal situation the reputation score would be based on the amount a peer's misbehavior and the amount of cooperation, however, often the amount a peers defects may be unknown [32]. For example it cannot be distinguished whether a peer dropped a message or whether the message was never received due to network failure. Moreover, the reputation score can be represented in numerous ways. Finally, multiple peers can be available to transact with and the scores should help the deciding peer select the best choice.

The response component of a reputation system which is probably the most crucial part, concerns itself with the actual creation of incentives. However, incentives that encourage cooperation may only work for selfish peers [32], therefore, punishment is required to mitigate other types of peers that do not cooperate (e.g. hackers that try to exploit the system). Incentives are naturally created through improved quality or quantity of service and alternatively additional payments can also come into play. Punishment on the other hand is usually in the form of exclusion from the network.

Approaches

As we have mentioned, there are many approaches to designing reputation systems and the choice is very much dependent on the application. Feldman et al. identify a family of more robust techniques that provide different trade-offs [19]:

- Local history: Each peer keeps a constant sized list of the both the peers it has successfully transacted with and peers that it has donated to and randomly chooses between the two categories for its next transaction.
- Shared history: A shared history infrastructure such as a Distributed Hash Table (DHT) is used to store the a global history. Peers access the global history to determine who to interact with.
- Maximum flow based subjective reputation: A maximum flow algorithm is applied to a global history to filter out false information and subjectively compute the reputation of a peer.
- Adaptive stranger policy: peers adapt their behavior to strangers according to their past experiences. This could leave little incentives for peers to change identity (whitewash).
- Short term history: transactions expire over time which creates a short term history. Short term histories prevent peers from building up reputation and turning into traitors later on.

Examples of Reputation Systems

EigenTrust is a popular example of a completely distributed global reputation system for P2P networks [26]. In EigenTrust each peer *i* is assigned a unique global trust value that reflects the past experiences of all the other peers in the network with peer *i*. All peers participate in computing these values in a distributed symmetric manner with some overhead on the network.

The idea behind EigenTrust is to expand local trust values to global scale weighted by the global reputations of the assigning peers [26]. That is, each peer i keeps track of his satisfaction level of transacting with peer j. The algorithm builds upon the idea that a peer i will have a high opinion of those peers who have been cooperating and is also likely to trust their opinions about other peer's reputations.

In their work, Kamvar et al. demonstrated that the idea behind EigenTrust leads to a system where global trust values correspond to the left principal eigenvector of a matrix of normalized local trust values. The normalization is done by computing the global reputations of a peer i given by the local trust values assigned to it by other peers, weighted by the global reputations of the assigning peers.

BarterCast in another example of a fully distributed reputation system based on ratios of upload and download in a P2P system and has been successfully implemented in Tribler [1]. The reputation system works by broadcasting interaction between peers to neighbors and using a Maxflow based algorithm to compute reputation scores [34].

The outline of the protocol is as follows: First, upload and download statistics of peers are spread in the network among peers. Second, these statistics are used by a peer to create a local view of the data transfer in the network. Third, based on this local view, a peer computes the reputation of other peers it encounters.

An epidemic protocol is used for peer discovery where peers exchange lists of random peers with all other peers they encounter. All such peers are periodically polled to check their connectivity and to transfer new information. In addition peers keep track of the data they have exchanged with other peers in the P2P system. Two types of statistic information are exchanged: (**i**) data on a peer's own exchanges with other peers (**ii**) data on exchanges between known peers and other third parties. As a result of such exchanges each peers creates a local view of the exchanges in the network.

The local views are aggregated according to a reputation metric which is defined as $upload - download$. A subjective reputation of a peer j at peer i is computed with a maximum flow algorithm. Finally scaling is involved to scale the relationship between the reputation value and flow.

For more details of the reputation systems readers are advised to refer to the original work of Kamvar et al. and Meulpolder et al.

While reputation systems can incentivize seeding beyond the TFT framework other popular approaches have also emerged that currently demonstrate significant success in providing incentives.

2.2.2 Private BitTorrent Communities

A second popular approach to incentivize seeding after a peer has finished downloading a file has emerged in private BitTorrent communities. Examples of private BitTorrent communities are TVTorrents and FileList.org. Membership in such communities is by invitation and it is relatively hard to join such communities.

Private BitTorrent communities use the sharing ratio of peers (i.e. ratio of the amount of upload against amount of download) as reputation scores.[4] Communities such as TV-

[4] This is referred to as sharing ratio enforcement.

Torrents require very high sharing ratios to maintain membership [33] and peers with low sharing ratios are deprived of their membership status. Other private communities such as FileList.org have a less strict membership policy. For example low sharing ratios in FileList.org only result in limited access to content (new content cannot be accessed) but membership can still be maintained.

In private BitTorrent communities the sharing ratio of each individual peer is computed and stored at the community tracker. This can be achieved through the regular status updates that peers send to the tracker. Therefore, the tracker can exercise control over which peers are allowed to transact with others and what content they can download.

Meulpolder et al. have demonstrated that peers of private BitTorrent communities show a much higher tendency to seed and that the ratio of seeders to leechers is much higher in private communities. Such properties result in high performance and fast download times [33].

2.2.3 Issues and Vulnerabilities

Even though reputation systems and private BitTorrent communities complement the BitTorrent protocol by incentivizing seeding beyond the TFT mechanism there are several issues that have lead us to defining our research goal as stated in Chapter 1. We elaborate on each approach respectively.

The main problem with reputation systems is that they are exposed to vulnerabilities ranging from simple lying to more complex collusion attacks [25]. Peers that have interacted with each other can lie about the amount of contribution that they have made to each other in order to gain higher reputation. Even though the maximum flow approach to reputation systems limits a peer's ability to lie [34] peers can still benefit from lying. On the other hand a group of peers could collude to lie about their reputation and divert the system into thinking that they have a high reputation score even with a max. flow approach.[5]

Another problem with reputation systems is the possibility of Sybil attacks [12]. Sybil attacks involve a single peer impersonating multiple identities and falsely claiming that it has transacted with those peers that are actually impersonations of the original peer. Sybil attackers can fool the reputation system into assigning them high reputation. Such peer can carry on their high reputation to improve future interaction. It has been shown that any global history reputation system that computes reputation scores non subjectively cannot be sybil-proof [8].

A third problem with reputation systems relates to the ability of peers to change identity with zero cost. This is referred to as whitewashing. Without sufficiently permanent identifiers reputation systems are practically easy to circumvent. Few BitTorrent clients implement such persistent identities.[6] Furthermore, without verification methods peers can also assume the identity of other peers and gain higher reputation scores by pretending to be others.

A fourth problem relates with the ability to launch distributed denial of service attacks (DDoS) attacks from reputation systems. That is, an attacker can assign a high reputation score to an node on the Internet that is not necessarily part of the P2P network. As a result,

[5] Collusion attacks are very difficult to implement and we will not consider such attacks in our research.

[6] Tribler is an example.

the victim would receive a large amount of requests that could lead to the degradation in quality of the service that the victim is actually meant to provide.

Finally there are problems related with the amount of communication that is required to implement reputation systems. Even though most popular reputation systems (i.e. EigenTrust and BarterCast) are distributed mechanisms, the amount of communication that is required to calculate reputation scores is high. This fact limits the scalability of these mechanism and trade-offs are usually made that reduce the accuracy of the reputation scores.

Regarding the second popular approach to incentivize seeding, the main problem with private BitTorrent communities is their exclusivity. That is private communities are hard to join and benefit a limited number of BitTorrent peers. A fair approach would consider improving quality of service for the larger public.

Secondly, sharing ratio enforcement in private BitTorrent communites is a centralized mechanism. That is all sharing ratios have to be computed and recorded at the community's tracker. As a result private BitTorrent communities are limited in scalability.

Finally private communities also suffer from discrimination against slower peers. That is a limited group of fast peers can gain high reputation scores and attract most of the potential interactions opportunities. As a result it would become very difficult for slower peers to maintain an acceptable sharing ratio (BitCrunch [24]). Peers that do manage to maintain an acceptable ratio would have to seed content for long periods even though they might have devoted all available resources to their peers and fully complied with the BitTorrent protocol.

Such problems reinforce the need to design incentive mechanisms that are scalable, fair and resistant to vulnerabilities and are the motivation behind this research. Having described our motivation we move on to providing background knowledge on some important concepts from game theory which are essential to understanding this research.

2.3 Game Theory

This section is an introduction to *game theory* which is a subfield of Economics. Its goal is to mathematically describe the decision making of players in strategic situations that are commonly described in the context of *games*. Game theory can provide a rich set of tools to model the P2P environment in order to gain better understating of the economic factors at play in the strategic decision making of peers within a P2P network. As bandwidth is a limited resource for peers, it is reasonable to assume that they would intend to 'spend' it wisely when interacting with other peers and therefore act strategically to preserve it.

2.3.1 Games and Strategies

Game theory can be described as the theory of social situations. It is relevant in describing many forms of human interactions *e.g.* games of poker, voting and essentially many situations where there is a conflict of interest between the parties that are involved. Interestingly, *games* are the tool with which game theory models social interactions. Consider the following example (one of the most well known examples in Game Theory) [36]:

P1 \ P2	Confess	Silent
Confess	4 4	5 1
Silent	1 5	2 2

Figure 2.2: The PD game cost matrix [36]

Example 2.3.1. ***(Prisoner's Dilemma (PD))*** *Two prisoners are on trial for a crime and each one faces a choice of confessing to the crime (C) or remaining silent (S). If they both remain silent, the authorities will not be able to prove charges against them and they will both serve a short prison term, say 2 years, for minor offenses. If only one of them confesses, his term will be reduced to 1 year and he will be used as a witness against the other, who in turn will get a sentence of 5 years. Finally, if they both confess, they both will get a small break for cooperating with the authorities and will have to serve prison sentences of 4 years each (rather than 5).*

The PD game is very well described by a matrix of incurred costs for each player; see Figure 2.2. Informally, game theoretic terminology refers to the choices of 'confessing' and 'remaining silent' as the prisoner's (*pure*) *strategies* or *actions* and their sentences are referred to as *costs*. Other games might involve *payoffs* for the players instead of costs. Interestingly, the PD game models situations that naturally arise in many social contexts including P2P networks, pollution, global warming and packet routing on the Internet.

Games such as the PD game can be represented in many forms, however the most commonly used form is the *normal-form* also known as the *strategic-form*. A game written in this way amounts to a representation of every players utility for every state of the world in the special case where states of the world depend only on the players combined actions.

It turns out that settings in which the state of the world also depends on randomness in the environment (called *Bayesian* games) can be reduced to (much larger) normal-form games [47]. Furthermore, games that involve an element of time, which are usually represented as *extensive-form games*, can also be represented in the normal form that sometimes lead an exponential number of possible states.

A natural way to represent games is via an n-dimensional matrix. We already saw a two-dimensional example in Figure 2.2. In general, each row corresponds to a possible action for the first player, each column corresponds to a possible action for the second player, and each cell corresponds to one possible outcome. Each players utility for an outcome is written in the cell corresponding to that outcome, with player 1s utility listed first.

Given the set of possible actions of a normal-from game, each player may choose to select actions according to his own *strategy*. That is, while one player's strategy for example, might be to select a single action (*pure strategy*), another player might choose to select a randomization over some actions. Such a randomized strategy is referred to as a *mixed* strategy.

Given a game and the player's strategies, game theory aims to analyze the utilities in the game from the perspective of a selfish utility maximizing player. For example, putting your self in the shoes of a prisoner in the PD game, game theory tries to answer questions such as: Given these options which choice should you adopt, C or S ? Does it depend on what you think your opponent will do? .... The answer to many of these questions is rooted in *solution concepts* of the game. We will present some solution concepts in detail.

2.3.2 Solution Concepts

Similar to a game of chess where a player is constantly deciding on his next best move a prisoner in the PD game is also trying to analyze his best move. The problem of choosing a next action or next strategy has a solution which is termed *solution concept*. Depending on the utility structure of the game, the player might find multiple solution concepts from which he can choose. Each of these solution concepts have their own characteristics and we will describe some of the most important ones in this section: *Dominant strategies* and *Nash equilibria*.

Dominant Strategies

Considering our example of the PD game from before, we can easily see that the only stable outcome of the game is that both prisoners confess. This is because in each of the other outcomes, at least one player can improve his utility by switching from 'silent' to 'confess' if he were given the chance to do so. This means that regardless of the strategies played by the other prisoner, each prisoner has a unique best strategy of 'always confess'. It is said that a game has a *dominant strategy* if it has this property.

Notice that the even though it is best for each player to always confess from a selfish view point, it is not the optimal outcome of the game. In fact, the optimal outcome in the PD game is that both prisoners remain silent. However, a very nice property of a dominant strategy solution is that if a game has such a solution, it is always chosen (under the assumption that players are selfish and utility maximizing). Therefore, dominant strategy solutions are a powerful tool in making behavior predictions.

On the other hand, having a single dominant strategy is a property that rarely occurs and many games do not have such a dominant strategy. This is because very few games actually have a best strategy of action regardless of the actions of the opponent(s). Fortunately, relaxing the independence of the solution from the opponent's strategies, yields an often found solution to a game which is referred to as a Nash equilibrium (in honor of its founder, *John Nash*).

Nash Equilibria

Many games do not have a dominant strategy solution since in many natural games the utility of a player is dependent on the strategy of other players. In such games we need to seek a less restrictive and more widely applicable solution concept. It has been proven that any game with a finite number of players and a finite number of strategies has a mixed strategy Nash equilibrium (Nash Theorem [36]). In fact, Nash equilibria have turned into the central most important solution concept in game theory and are important to understanding our research.

To define Nash equilibria more formally, we must first explain a notation that we will use from this point on. For a strategy vector $s \in S$ we use s_i to denote the strategy played by player i and s_{-i} to denote the $(n-1)$-dimensional vector of the strategies played by all other players. We will use $u_i(s_i, s_{-i})$ to denote the utility of player i in this case. Using this notation we have that [36]:

Definition 2.3.1. *(**Nash equilibrium**) A strategy vector $s \in S$ is said to be a Nash equilibrium if for all players i and each alternate strategy $s'_i \in S_i$, we have that*

$$u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i}). \tag{2.1}$$

In a Nash equilibrium no player i can change his strategy from s_i to an alternative s'_i in order to improve his utility assuming that all other players play their previously chosen strategies. Notice that Nash equilibria are stable and self enforcing. That is, it is in every player's best interest to stick to their strategy. Furthermore, It is clear that a dominant strategy is also a Nash equilibrium. That is, $DominantStrategies \subset NashEquilibria$. For example in the PD game the dominant strategy of 'always confess' is also a Nash equilibrium of the game. In fact, it is also the unique Nash equilibrium of the game.

Another variant of the Nash equilibrium is the solution concept of a Bayesian Nash equilibrium. Bayesian Nash equilibria are of importance when players do not have complete information on the other players and rather hold beliefs about their opponents types. In such settings players are assumed to choose strategies based on their type. Players maximize their *expected utility* in Bayesian settings given their beliefs. P2P networks are similar to the settings in which Bayesian Nash equilibria are relevant.

At this point we have introduced the essential concepts of game theory that are useful for understanding our research. In our next and final section we move on to providing some design principles that have guided us during our research. These principles are concerned with the complexity of a mechanism and are used as criteria to evaluate the complexity of our solution.

2.4 Mechanism Design

Many powerful tools and guidelines for designing exploit-proof mechanisms can be found within the frameworks of Mechanism Design (MD). MD is a subfield of economics with a rather unique engineering perspective. It deals with designing systems so that certain system wide properties (e.g. efficiency, fairness, stability) emerge in equilibrium from the constituents interaction[11]. Many MD principles make assumptions regarding the existence of a trusted central authority that can computationally be used to arrive at the desired properties. Nevertheless, many principles in MD lead to this center having to solve NP-complete problems and are practically difficult to use.

On the other hand, a subbranch of MD referred to as Algorithmic Mechanism Design (AMD) assumes that the center of trust is *rationally bounded* and applies algorithmic techniques such as randomization and approximation to avoid trying to solve NP-complete problems. In comparison AMD principles make more realistic assumptions and are more useful for designing mechanisms in real world settings. Nevertheless, AMD makes the assumption that connections between the players of a game and the trusted center are reliable. Furthermore, most work in AMD assumes that the players of the game are static, *i.e.* do not change over time [17, 11]. Such assumptions are not realistic in a P2P network and designing mechanisms for P2P networks requires different assumptions and notions.

Distributed Algorithmic Mechanism Design (DAMD) is a subfield of MD which is most consistent with the properties of P2P networks. The challenge in DAMD is that in distributed mechanisms the same players that seek to manipulate a system will also run the mechanism. While in centralized approaches players can only effect the outcome by choosing different strategies, in distributed settings players can actually effect the computation since its no longer computed by a single trusted entity. Clearly, this opens up numerous exploiting opportunities and is one of the main differences between AMD and DAMD. In DAMD distributed trust plays a vital role as players must now trust the computations performed by others.

On the other hand complexity in AMD and DAMD are inherently of a different nature [37]. That is if one could assume that the interconnection network between players is trustworthy then the main difference of AMD and DAMD would be the measure of complexity. It is widely agreed upon that any measure of complexity for distributed mechanisms should take into account at least the following quantities [37, 41, 17, 11, 16]:

- The total number of message sent over the network.
- The maximum number of messages send over any single link in the network.
- The maximum size of a message
- The local computational burden on each node.
- The storage required at each node.

Due to the large number of possible exploits in a distributed setting and the assumption that players act strategically, it is very important to provide incentives that ensure correct computation [17] while at the same time limiting the complexity to a manageable amount. This means that ideally, the total number of messages sent over the network would be linear in the number of peers in the network, and the total number of messages sent over a single link would be constant to avoid bottlenecks[17]. Traditional complexity measures apply to local storage and computational requirements. In our research we use the notion of complexity in a distributed setting as a guideline and we use the complexity measures listed above as criteria to compare solutions.

At this point we have provided the necessary background information required to understand the remainder of this text. In the following chapters we will present our findings and our research step by step. The next chapter is dedicated to our first step which is a measurement study of the FileList.org private BitTorrent tracker.

Chapter 3

Feasibility of Indirect Interaction

In chapter 2 we explained that BitTorrent's TFT mechanism does not incentivize seeding since seeders do not require pieces of a file which they already posses. Nevertheless, seeders might be leeching other content simultaneously. This means that even though seeders do not require pieces of the file that they are seeding they might still be interested in acquiring pieces of another file. This is what we refer to as indirect interaction. This chapter is dedicated to measuring the feasibility of such a scenario. Intuitively this scenario might seem quite rare, however we speculate that it should be quite possible for seeders to interact indirectly based on other studies [22]. We investigate the correctness of this speculation in the logs of the FileList.org private BitTorrent tracker. FileList.org is a relatively small community of some $110,000$ *members and contained approximately* 3000 *files that were bartered at the time the logs were gathered.*

The chapter is organized as follows. In Section 3.1 We present our motivation behind conducting our measurement study of filelist.org and the reasons for which we initially speculated indirect interaction to be possible. In Section 3.2 we present some important details of our filelist.org data set. In Section 3.3 we elaborate on how we have conducted our measurement study and present formal definitions of indirect interaction and multi-swarming. Additionally, we provide some details on the data structures and algorithms used to accomplish this task. Subsequently, we present the results of our measurements in Section 3.4. Finally we end this chapter with a reflection on what difficulties we have faced, reasons behind our choice of FileList.org as our data set and what could be done differently in the similar future measurements.

3.1 Motivation

One of the key differences between BitTorrent and previous generation of P2P file-sharing applications is that unlike previous P2P systems that are based on peers exchanging different files with each other, BitTorrent is based on peers exchanging pieces of a single file with each other. This difference results in direct interaction rather than indirect interactions in which several peers would possibly need to exchange files in order to obtain the files that they actually seek to acquire.

While some measurement studies of BitTorrent suggest that direct interactions based on bartering bandwidth with a Tit-for-Tat mechanism are actually the basis for the high performance and bandwidth utilization of BitTorrent [10, 45, 52, 28, 43], other studies suggest that a more strict *piece for piece* Tit-for-Tat strategy improves fairness while still able to match the performance of the current protocol [13, 6]. But in general, there seems to be a consensus on the merits of direct interactions and there are only discussions on the type of strategies and implementation employed for direct interaction. However, if we are to go with the same consensus we are still left with an open question of what actually incentivizes *free seeding of content* in BitTorrent. That is, seeders are only characterizable as altruistic peers within this framework. As explained before this constitutes the main problem of P2P architectures. Our motivation for conducting a measurement study is to find possible ways of going beyond a dependency on altruistic behavior.

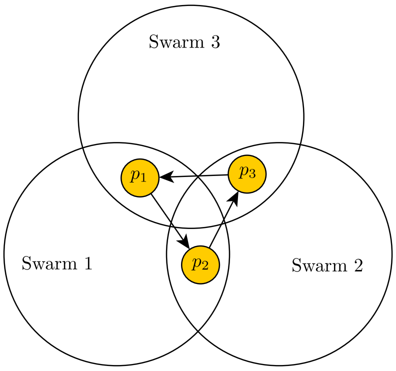
While seeders might not be in need of pieces of a file which they already posses they might still be in need of pieces of different files. Studies show that many peers are actually actively seeding/leeching multiple files simultaneously (multi-swarming) [22] and hint at the possibility of satisfying the demands of seeders that multi-swarm. This can be achieved through indirect interaction.

3.1.1 The Structure of Indirect Interaction

In real world settings, regardless of the parties and the setting , direct bartering transactions are only possible when supply and demand match (from a closed loop), unless one party is a pure altruist and is willing to give away things for free. For example consider two merchants a and b. Merchant a requires rice and is willing to give wheat in return for rice. Merchant b has rice to give away but has no use for wheat and will only accept potatoes in return for his rice. Therefore, a direct transaction between these two merchants is not possible. The problem is to find a solution to make a transaction between a and b possible. One such solution is *indirect interaction*. Assume that there is a third merchant c (the key to indirect interaction) who has potatoes to give away and is willing to accept wheat. The addition of the third merchant creates a possibility for a 3-way transaction and allows for merchants a, b and c to successfully interact indirectly.

The merchant example is an analogy of a situation that BitTorrent seeders find themselves in. Nevertheless, we already know that a large number peers are active in multiple swarms. Such peers could possibly play the role of the key to indirect interaction. An interesting and similar situation to the merchants example is therefore imaginable for multi-swarming peers that are seeding in at least one of the swarms they are participating in:

Example 3.1.1. *(Indirect Interaction of 3 Peers) Consider three peers* $\{p_1, p_2, p_3\}$ *and three swarms* $\{1,2,3\}$ *where* p_1 *is seeding in Swarm* 1 *and is leeching in Swarm* 3*,* p_2 *is seeding in Swarm* 2 *and is leeching in Swarm* 1 *and* p_3 *is seeding in Swarm* 3 *and is leeching in Swarm* 2 *(See Figure 3.1). These peers have exactly the same situation as in the merchants example from before. The content that peers seed is equivalent to what the merchants are willing to supply and the content that peers leech is equivalent to what the merchants demand.*



Swarm	Peers	Peer Type
1	p_1 p_2	Seeder Leecher
2	p_2 p_3	Seeder Leecher
3	p_3 p_1	Seeder Leecher

Figure 3.1: Example of the indirect interaction of 3 multi-swarming peers.

In Example 3.1.1 the three peers are able to do an indirect interaction by p_1 seeding p_2, p_2 seeding p_3 and p_3 seeding p_1 in return. This example demonstrates an important point about the incentive structures of BitTorrent. While currently seeders exist in BitTorrent swarms for probably altruistic reasons, indirect interaction (and hence Indirect Reciprocity) could very well provide the incentives for all three peers $\{p_1, p_2, p_3\}$ to seed content, at least while they are still leeching in other swarms. Such a possibility clearly describes our motivation for our first step in this research; a measurement study that would indicate how probable it is for indirect interactions to occur.

We already know that the BitTorrent protocol only supports direct interactions. To enable the indirect interaction of peers, BitTorrent requires an extension that allows peers to find each other and barter pieces of multiple files instead of pieces of the same file.

To clarify how such a protocol could work, consider the following example:

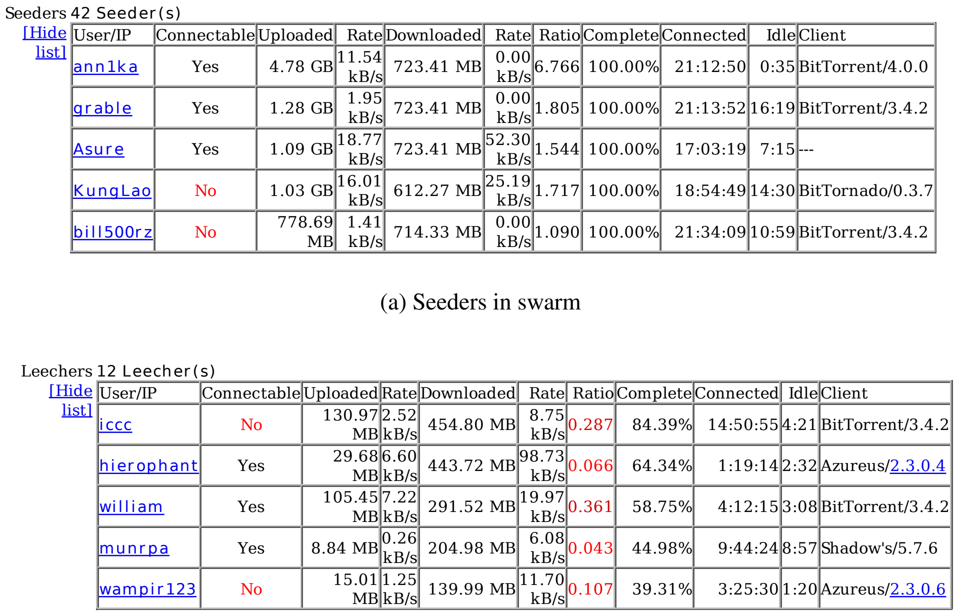
Example 3.1.2. *(A Centralized Extended BitTorrent Protocol) Upon joining a swarm, each peer declares to the tracker of that swarm all other swarms in which it is participating. The tracker(s) of all swarms use this information about their multi-swarming peers as follows: If indirect interactions between multi-swarming peers is possible, the trackers will coordinate those peers to connect to each other and start bartering file pieces.*

In chapter 4 we will discuss the design of such a protocol in detail. What we would like to find out for now is the chances that each multi-swarming peer has to interact with other multi-swarming peers in the framework of indirect interaction. In other words, we are investigating the feasibility/practicality of such an extension to the BitTorrent protocol. But before we dive into more detail let us elaborate on the FileList.org data set which intend to use as the basis for our measurements.

3.2 Our Data Set: The *FileList.org* BitTorrent Community

The *FileList.org* tracker log is a data set of user level information from the *filelist.org* private BitTorrent community which has been previously compiled and studied by Roozenburg et al. [46]. The data set has been logged for a period of several months over late 2005/beginning 2006 and at the time of the study the filelist community consisted of a member base of some 110,000 people. Every member of the filelist community is given a unique username and the community has a limit on the number of members. The number of user accounts is believed to have reached the limit at the time the data set was gathered.

Whilst the filelist tracker performs the standard functionalities of a BitTorrent tracker it also keeps data of all activity based on the username of each peer. The information gathered by the tracker is used to determine each peers' *sharing ratio* (see Figure 3.2). The sharing ratio of each peer determines if it meets the membership standards which are determined by the policies of the community (*Sharing Ratio Enforcement*). Peers that do no meet these standards are denied access to some of the content. As a result of the sharing ratio enforcement, the FileList.org community tends to be one in which peers are more cooperative in comparison with public BitTorrent trackers such as thePirateBay.org .[1] On the other hand, keeping track of the sharing ratios of all users is a complex task for the filelist tracker therefore, to reduce the work load of the tracker, the community policy is to remove torrents 28 days after their creation [46].



Seeders 42 Seeder(s)

[Hide list]

User/IP	Connectable	Uploaded	Rate	Downloaded	Rate	Ratio	Complete	Connected	Idle	Client
ann1ka	Yes	4.78 GB	11.54 kB/s	723.41 MB	0.00 kB/s	6.766	100.00%	21:12:50	0:35	BitTorrent/4.0.0
grable	Yes	1.28 GB	1.95 kB/s	723.41 MB	0.00 kB/s	1.805	100.00%	21:13:52	16:19	BitTorrent/3.4.2
Asure	Yes	1.09 GB	18.77 kB/s	723.41 MB	52.30 kB/s	1.544	100.00%	17:03:19	7:15	---
KungLao	No	1.03 GB	16.01 kB/s	612.27 MB	25.19 kB/s	1.717	100.00%	18:54:49	14:30	BitTornado/0.3.7
bill500rz	No	778.69 MB	1.41 kB/s	714.33 MB	0.00 kB/s	1.090	100.00%	21:34:09	10:59	BitTorrent/3.4.2

(a) Seeders in swarm

Leechers 12 Leecher(s)

[Hide list]

User/IP	Connectable	Uploaded	Rate	Downloaded	Rate	Ratio	Complete	Connected	Idle	Client
iccc	No	130.97 MB	2.52 kB/s	454.80 MB	8.75 kB/s	0.287	84.39%	14:50:55	4:21	BitTorrent/3.4.2
hierophant	Yes	29.68 MB	6.60 kB/s	443.72 MB	98.73 kB/s	0.066	64.34%	1:19:14	2:32	Azureus/2.3.0.4
william	Yes	105.45 MB	7.22 kB/s	291.52 MB	19.97 kB/s	0.361	58.75%	4:12:15	3:08	BitTorrent/3.4.2
munrpa	Yes	8.84 MB	0.26 kB/s	204.98 MB	6.08 kB/s	0.043	44.98%	9:44:24	8:57	Shadow's/5.7.6
wampir123	No	15.01 MB	1.25 kB/s	139.99 MB	11.70 kB/s	0.107	39.31%	3:25:30	1:20	Azureus/2.3.0.6

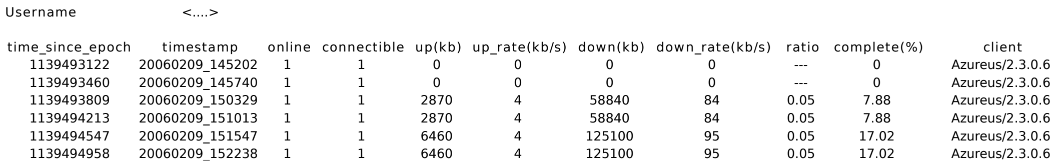
(b) Leechers in swarm

Figure 3.2: Extract from the FileList.org tracker logs for a single swarm

Roozenburg et al. have monitored the tracker log files (See Figure 3.2) of every available swarm over the period of 2005-2006 using a data crawler and timestamped the logs to create detailed information about the peers that were online in every swarm, their type,

[1] For a more specific details on the differences between private and public BitTorrent trackers please refer to the work of Meulpolder et al. [33].

sharing ratio, amount of download and upload, *etc.* This detailed information, which has a granularity in the order of several minutes per piece of information, has been formatted as the data presented in Figure 3.3 and forms the basis for our measurement of the probability of indirect interaction between multi-swarming peers. Since the trace files are based on usernames and not on IP addresses they are perfectly suited for finding out which peers have been online and in which swarms they have been active. The uniqueness of usernames guarantees that information about peers is accurate. The data on each peer also includes what their status has been at each moment (i.e. leecher or seeder). On the other hand, detailed information about the pieces of the file that peers posses at each moment is missing, but as we will explain later (See Section 3.3) this will not create a problem in our analysis of peer behavior.



Username <....>

time_since_epoch	timestamp	online	connectible	up(kb)	up_rate(kb/s)	down(kb)	down_rate(kb/s)	ratio	complete(%)	client
1139493122	20060209_145202	1	1	0	0	0	0	---	0	Azureus/2.3.0.6
1139493460	20060209_145740	1	1	0	0	0	0	---	0	Azureus/2.3.0.6
1139493809	20060209_150329	1	1	2870	4	58840	84	0.05	7.88	Azureus/2.3.0.6
1139494213	20060209_151013	1	1	2870	4	58840	84	0.05	7.88	Azureus/2.3.0.6
1139494547	20060209_151547	1	1	6460	4	125100	95	0.05	17.02	Azureus/2.3.0.6
1139494958	20060209_152238	1	1	6460	4	125100	95	0.05	17.02	Azureus/2.3.0.6

Figure 3.3: Extract from the timestamped user level information for and instance of a swarm and a peer (anonymized)

Before we actually present the results of our measurements and their analysis we need to present some notions and data structures that have been used to conduct the measurement.

3.3 Measurement Method and Definitions

In order to study the feasibility of indirect reciprocity between multi-swarming peers we have used several notions and data structures that are based on the principal of finding the supply and demand of file pieces between multi-swarming peers as demonstrated in Example 3.1.1. Since our data set is missing detailed information about the actual file pieces that each peer possesses over time, we have made the assumption that multi-swarming peers remain interested in bartering with each other at all times. This assumption is actually justifiable because of the bartering policies of the BitTorrent protocol. The policy by which pieces of a file are spread between peers in a BitTorrent swarm ensure sufficient entropy to allow us to assume that peers will remain interested in the pieces of the file that other peers posses at all times [29]. Knowing that the level of detail of the data is not a hindrance, we can now discuss the notions that our measurements are based on.

The first notion that we have based our measurements on is the definition of an *indirect interaction.* Up to this point, we have not clearly defined what we mean by indirect interaction and have not defined the type of indirect interactions that we aim to measure. The definition of Indirect Interaction is quite intuitive and simply stated includes every type of interaction that is not a direct interaction. let us first define what we mean by direct interaction:

Definition 3.3.1. ***(Direct Interaction)*** *An interaction between BitTorrent peers is consid-*

ered to be ***direct*** *if it takes place between two peers within the same swarm, and the piece(s) that are exchanged between the parties that are involved belong to the file that corresponds to that swarm.*

Note that this definition includes the standard *Tit-for-Tat*, *Optimistic Unchoke* and *Seeder Unchoking* interactions of BitTorrent peers. Therefore, any type of interaction that is possible in the current BitTorrent protocol is considered to be a direct interaction. Given this definition of direct interaction we can define indirect interaction by

Definition 3.3.2. ***(Indirect Interaction)*** *An interaction between BitTorrent peers is considered to be* ***indirect*** *if it is not a direct interaction.*

Note that this definition includes interactions of the type shown in Example 3.1.1.

While the definition of indirect interaction covers many types of interactions between peers not all of these interactions are interesting in our measurement study. To be precise, we are only interested in the types of indirect interactions that form a closed loop of supply and demand between the involved parties. The choice of a closed loop for indirect interaction seems natural because it is able to match the supply and demand and hence able to create the incentive for parties to actually get involved without having to resort to altruistic behavior. Therefore, closed loop indirect interactions are what we are looking for. We also refer these interaction as cyclic interactions.

A second important notion is the definition of a *multi-swarming* peer. Up to this point we have implicitly used a loose definition of '*peers that are online in more than one swarm at a moment in time*' to describe multi-swarming peers. In order to give a more formal definition of multi-swarming we need to consider that peers that are online in more than one swarm but are not leeching content in any of the swarms do not actually demand any file pieces. Such peers are again only characterizable as altruistic peers. Therefore, a more precise definition of multi-swarming would need to take this fact into account. Hence we define a multi-swarming peer as:

Definition 3.3.3. ***(Multi-Swarming Peer)*** *A peer p is a* ***multi-swarming peer*** *if it is online in Swarms $S = \{s_1, s_2, \dots\}$ where $|S| \geq 2$ and $\exists s_i \in S$ in which p is a leecher.*

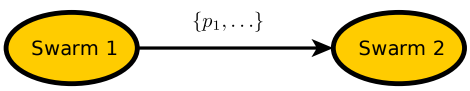
Using the definitions of indirect interaction and multi-swarming peers we have created a directed graph of the supply and demand between swarms to measure the possibility of indirect reciprocation.

3.3.1 Directed Graph of Supply/Demand

In order to create a structure for analyzing the possible interactions between multi-swarming peers we have created a data structure that is analogous to having a global view of swarms from the view point of a BitTorrent tracker. This data structure is a directed graph in which swarms are the nodes in the graph and the directed edges represent the possible indirect interactions between multi-swarming peers.

To create this graph, we have used a set of conventions based on definitions 3.3.2 and 3.3.3 to guide its creation. These rules can be stated as follows:

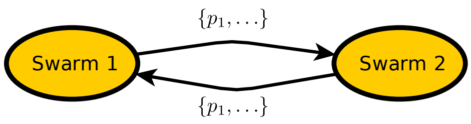
Convention 1. ***(Seeder-Leecher)*** *Consider two swarms labeled as Swarm*1 *and Swarm*2 *in the directed graph G. There is a directed edge from Swarm*1 *to Swarm*2 *in G if there exists a peer p that is a leecher in Swarm*1 *and a seeder in Swarm*2 *(See Figure 3.4). The edge is labeled with a set S for which* $p \in S$*.*



Swarm	Peers	Peer Type
1	p_1 ⋮	Leecher ⋮
2	p_1 ⋮	Seeder ⋮

Figure 3.4: Convention-1 for the creation of Supply and Demand Graph.

Convention 2. ***(Leecher-Leecher)*** *Consider two swarms labeled as Swarm*1 *and Swarm*2 *in the directed graph G. There is a directed edge from Swarm*1 *to Swarm*2 *and a directed edge from Swarm*2 *to Swarm*1 *in G if there exists a peer p that is a leecher in both Swarm*1 *and Swarm*2 *(See Figure 3.5). Each edge is labeled with a set S for which* $p \in S$*.*



Swarm	Peers	Peer Type
1	p_1 ⋮	Leecher ⋮
2	p_1 ⋮	Leecher ⋮

Figure 3.5: Convention-2 for the creation of Supply and Demand Graph.

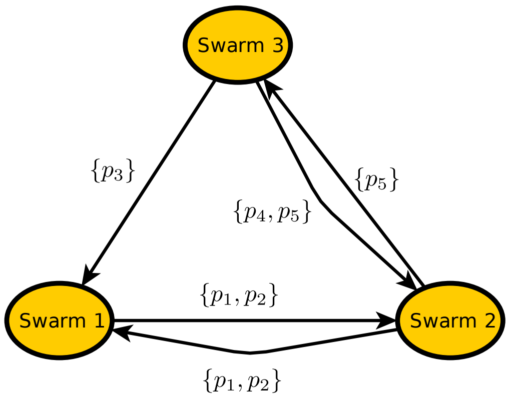
Convention 3. ***(Seeder-Seeder)*** *Consider two swarms labeled as Swarm*1 *and Swarm*2 *in the directed graph G. If there exists a peer p that is a seeder in both Swarm*1 *and Swarm*2 *there is no edge between Swarm*1 *and Swarm*2 *unless one of the previous conventions applies. If any of the previous conventions apply,* $p \notin S$ *for any of the possible label sets* S*.*

Convention 4. ***(No Multiple Edges)*** *If either of the previous conventions 1 or 2 apply multiple times for the same two swarms, only a single instance of each possible edge is created and peers that create this multiple application are simply added to the corresponding label sets.*

Note that Convention 1, 2 and 3 are directly related to our definition of a multi-swarming peer (Definition 3.3.3). Also note that the creation of edges in the graph which correspond to possible interactions of peers have been based on the assumption that we made about peers remaining interested to interact at all times (see beginning of this section). Convention 4 however, is only to keep the data structure that we have created simple.

To further clarify the use of these conventions we will provide the reader with an example of how they have been used.

Example 3.3.1. *Consider the set of swarms and peers depicted in Figure 3.6. In this example peer* p_1 *is a leecher in both swarms* 1 *and* 2 *and therefore Convention 2 applies to this peer. Therefore we have created two directed edges between swarms* 1 *and* 2*. Similarly* p_2 *is also a leecher in both swarms, however, Convention 4 has been applied and* p_2 *has only been added to the edge label sets. On the other hand peer* p_3 *is a seeder in Swarm* 1 *and a leecher in Swarm* 3*. Convention 1 applies to this peer and a single edge from Swarm* 3 *to* 1 *is created in the graph. Finally consider peer* p_6 *that is a seeder in both Swarm* 1 *and* 3*. According to Convention 3 this peer does not modify the edges between Swarm* 1 *and* 3*.*



Swarm	Peers	Peer Type
1	p_1 p_2 p_3 p_6	Leecher Leecher Seeder Seeder
2	p_1 p_2 p_4 p_5	Leecher Leecher Seeder Leecher
3	p_3 p_4 p_5 p_6	Leecher Leecher Leecher Seeder

Figure 3.6: Example of a Supply and Demand graph.

We will now explain how the graph of supply and demand can be analyzed to measure the probability of indirect interaction between multi-swarming peers (assuming that other multi-swarmers are willing to reciprocate).

3.3.2 Analysis of a Supply/Demand Graph

Having explained that we are interested in interactions that form a closed loop between peers (we will use the word loop and cycle interchangeably through out this chapter), it should be fairly simple to see how the graph of supply/demand is able to help us find closed loop indirect interactions. That is, such interactions appear in the form of cycles in the graph of supply/demand. For example consider the supply/demand graph of Example 3.6. Clearly, a cycle of the form $Swarm1 \rightarrow Swarm2 \rightarrow Swarm3 \rightarrow Swarm1$ exists in this graph. This cycle demonstrates that peers that are members of the label sets along these edges of the cycle have a possibility of indirect interaction. The only simple rule that applies here is that if at each edge of the cycle we choose a peer from the label set of that edge, this peer cannot be selected in the following edges. In our example peers $p_1 \rightarrow p_5 \rightarrow p_3 \rightarrow p_1$ can do a 3-way transaction of file pieces where p_1 can provide pieces of the file in *Swarm*2 to p_5, p_5 can provide pieces of the file in *Swarm*3 to p_3 and p_3 can provide pieces of the in *Swarm*1 to p_1. Notice, that we do not need these peers to be seeders and that leechers can also provide pieces of the files to each other as described.

For our measurements we have developed code that does the same analysis on the supply/demand graph by finding the cycles in the graph. Our measure of feasibility has been the probability of a multi-swarming peer being in a cycle. This measure of feasibility directly indicates the chances that a multi-swarming peer has at reciprocation (again assuming that peers remain interested in each other).

To calculate the probability of a multi-swarming peer being able to interact indirectly in a cycle, all we need to do is to divide the number of multi-swarming peers that form closed loop interactions by the total number of multi-swarming peers.[2] That is

Definition 3.3.4. ***(Probability of Reciprocation)***

$$\boldsymbol{Prob.} = \frac{|\{p : p \textit{ is multi-swarming and is in cycle}\}|}{|\{p : p \textit{ is multi-swarming}\}|}$$

Even though this might seem like a simple task, let us elaborate on the complexity of the task at hand before we proceed with our measurement results. One problem with this approach to calculating the feasibility is that it requires us to find all cycles within a directed graph. This problem can be described as:

Definition 3.3.5. *(#CYCLE) Is the problem of computing, given a directed graph G, the number of simple cycles in G. (A simple cycle is one that does not visit any node twice except for the starting node.)*

Unfortunately, it is known that *#CYCLE* is $\#P-Complete$. Let us elaborate on this class of problems. In an *NP* decision problem, we ask the question of whether there *exists*

[2] Our code and algorithms for performing this task can be on the Tribler SVN repository [1].

a solution to a given instance. However, for some problems, such as the one we are faced with, we are interested to know the *number* of solutions. Such problems are classified in *complexity theory* as the class #*P*. This class can be viewed as the equivalent of the class *NP* for decision problems. The class $\#P - complete$ can be loosely defined as, counting problems for which the existence of a polynomial time algorithm will imply that $\#P = FP$ where FP is the equivalent of the class *P* for counting problems [5].

In order to avoid the complexity problem that we have just described, we have made some simplifications in our measurements which are justifiable as follows. Since cycles in our directed supply/demand graph represent indirect interactions between multi-swarming peers, the longer the cycle the greater the number of peers that are involved. Since the peers that form a cycle are required to be online at the same time to interact longer cycles have more strict requirements with respect to peers being online. That is, more peers have to be online at the same time. However, we expect it to be more improbable to find peers that are online at the same time as the length of the cycle grows. This simple expectation allows us to limit the cycles that we are looking for to a practical length. In our measurements we have limited this length to 3 and for some instances to 5.[3] We have computed the probability of indirect interaction separately for each cycle length to also have an understanding of what a practical cycle length is in practice. We follow with our measurement results in the next section.

3.4 Measurement Results

We have intuitively divided our probability measurements into two categories of ***i***) A small random sample of 101 Swarms, ***ii***) 2972 Swarms in the data set. This intuition has risen from our initial expectations of what we would find from the analysis of the data set. That is, we expect to have very low probability of indirect interaction with knowing little or relatively close to nothing about supply and demand (*i.e.* only knowing about 101 swarms out of 2972 available ones) and higher probability when there is an abundance of information on supply and demand (*i.e.* knowing about 2972 swarms). Even though these categories lie on two extreme sides of a spectrum, we find that the results are quite useful and significant for reasons that we will explain later on. For both of the categories, we have measured the probability of indirect interaction every 1 hour beginning form Feb. 2006 to end Feb. 2006.[4] Therefore we have information on the probability for indirect interaction during every hour for the duration of almost a month.[5] We present each set of results respectively.

3.4.1 Random Sample of 101 Swarms

In this category of measurement, we have chosen a random sample of 101 swarms from the total 2972 swarms available in our data set.The sample includes all swarms that have a hash code starting with the letter ‘a’. These happen to add up to 101 swarms. This would be

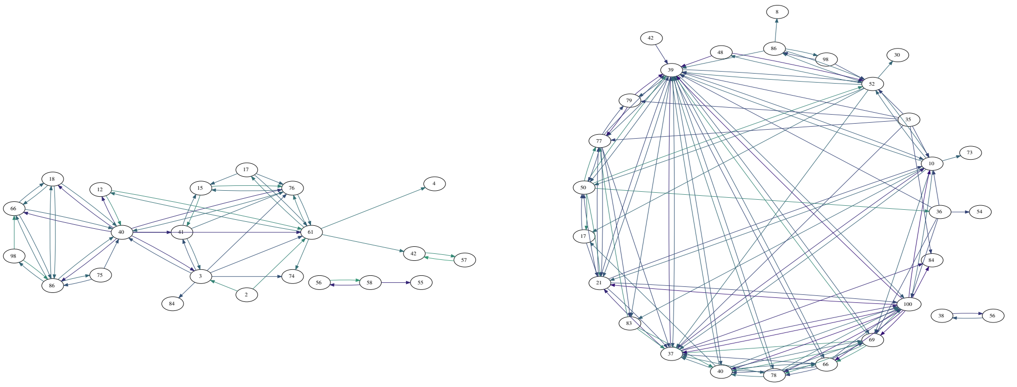
[3] This is similar to Fixed Parameter Tractable Algorithms for solving NP-Complete problems.

[4] The data set ends at 23rd of Feb.

[5] At some points during the period of measurements data seems to be missing due to failure.

equivalent to our BitTorrent protocol extension only having access to information about 101 swarms and the peers that are in them (or only keeping track of the multi-swarming peers in 101 swarms). Nevertheless, another 2871 swarms are still present and tracked according to the standard BitTorrent Protocol but our extended protocol does not know about them. As we explained before, this amount of information tracking lies on a very extreme side of a spectrum, however, it can indicate what our extended protocol is potentially capable of when given little information.

In order to get a feeling of how an actual supply/demand graphs looks like we have included two examples in Figure 3.7. This figure visualizes the supply/demand Graph for two instances of low probability of indirect interaction and high probability. For the first instance we have a total of 25 lonely peers and 36 multi-swarming peers while there are 4 cycles of length 2, 9 cycles of length 3, 8 cycles of length 4 and 12 cycles of length 5. The second instance actually has 99 multi-swarming peers with 25 lonely peers and a total number of 21 cycles of length 2, 43 cycles of length 3; 112 cycles of length 4 and 269 cycles of length 5.



(a) 2006-02-05 06:00:00.

(b) 2006-02-19 06:00:00.

Figure 3.7: Examples of Supply/Demand Graph: (a) Low probability of indirect interaction, (b) High probability of indirect interaction.

By analyzing the supply/demand graphs (one per every hour during the period that we mentioned earlier) we have calculated the probability of reciprocation according to Definition 3.3.4. The results demonstrated in Figure 3.8 depict the probability of a multi-swarming peer from the 101 swarms interacting in cycles with other multi-swarming peers per each cycle length (see Definition 3.3.4). Furthermore, Figure 3.9 gives some statistics about the total number peers that were online, multi-swarming peers that were online and the portion of those that are *lonely*. Lonely peers are multi-swarming peers that cannot participate in an indirect interaction loop of length 2. That is, these peers are either the only multi-swarming peer between two swarms or, other peers also multi-swarm between the same two swarms but their combination does not create edges in both direction in the supply/demand graph to form a cycle. For example, peer p_3 from Example 3.6 is a lonely peer. Lonely peers are an indication of the multi-swarming peers that are more difficultly able to form cycles with other peers, because as it turns out, many peers are able to form cycles of length 2 and higher lengths.

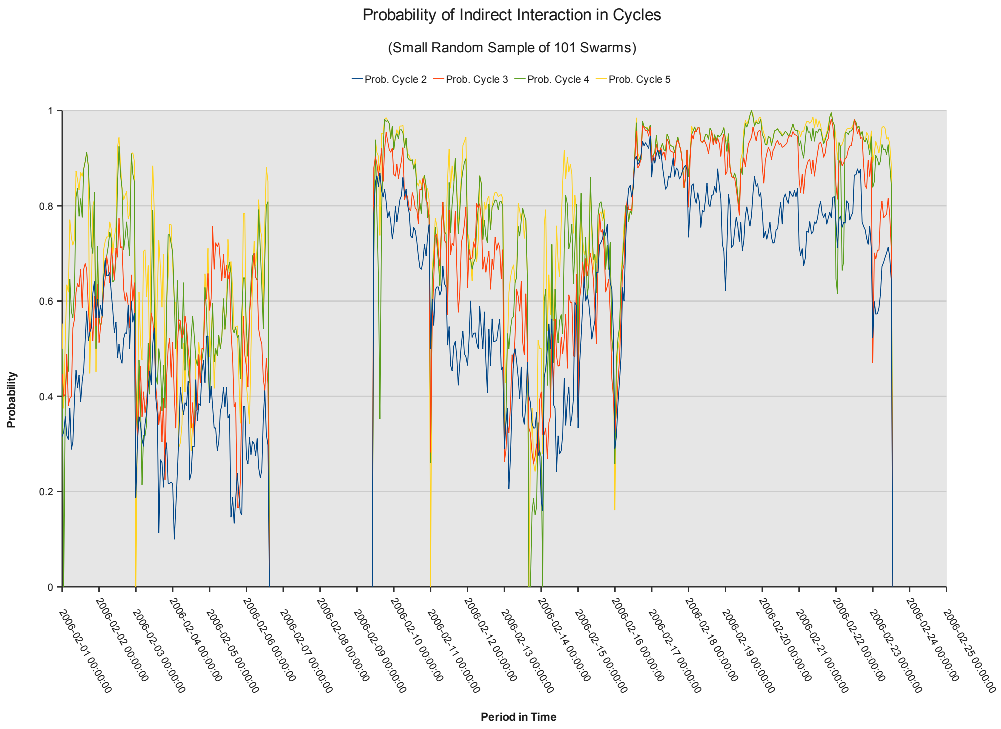


Figure 3.8: Probability of multi-swarming peers forming cycles of length 2 to 5.

Notice that the peer statistics show a low number of multi-swarming peers (See Figure 3.9). This is related to the fact that we have only considered a small number of swarms for our measurement. Even though at times we have a large number of online peers many of these peers might be multi-swarming in other swarms that are not included in the small sample. As a result they will not be detected as multi-swarming peers given the small sample by the extended protocol.

Even though our peers statistics (Figure 3.9) show a low number of multi-swarming peers, the probability of forming cycles is found to be surprisingly significant (see Figure 3.8). This suggests that even though multi-swarming peers are not significantly numerous in proportion to the total number of online peers, indirect interaction is still possible. This indicates that tracking a small number of swarms, still allows multi-swarming peers to interact indirectly with acceptable probabilities.

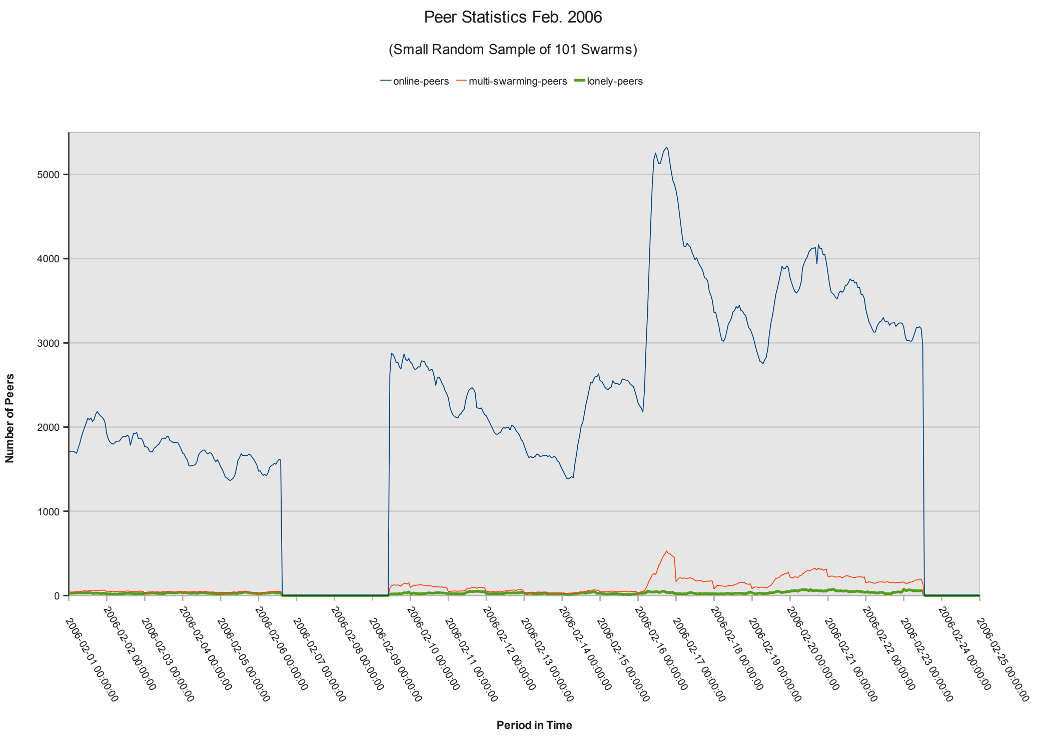


Figure 3.9: Statistics on the multi-swarming peers of the Small 101 Sample.

3.4.2 Large Sample of 2972 Swarms

In this category of measurement, we have included 2972 swarms of the data set in our measurements which includes almost all of the swarms in the data set.[6] This would be equivalent to our BitTorrent protocol extension tracking almost all multi-swarming peers in the entire the FileList.org tracker.

As before, we have created supply/demand graphs, one per every hour during the period of measurement for all swarms. In the case of our large sample it is more difficult to visualize the supply/demand graphs as we did for the small sample. Instead Figure 3.10 depicts a single instance of the supply/demand graph adjacency matrix. Each pixel sized column/row in Figure 3.10 corresponds to a swarm. Pixels that are black depict that there is no edge from the swarm in the corresponding row to the swarm in the corresponding column. On the other hand colored pixels depict the existence of an edge. We can derive that there is a high number of edges in the supply/demand graph as there are a large number of colored pixelsin the figure. Notice the relative diagonal symmetry in Figure 3.10. Diagonal symmetry in the adjacency matrix indicates the existence of cycles of length 2. In fact the

[6] We have excluded a small number of swarms in the data set that contain only one seeder from the measurement due to space limitations.

number of feasible cycles of length 2 in figure 3.10 turns out to be 2649 while there are 81148 feasible cycles of length 3.

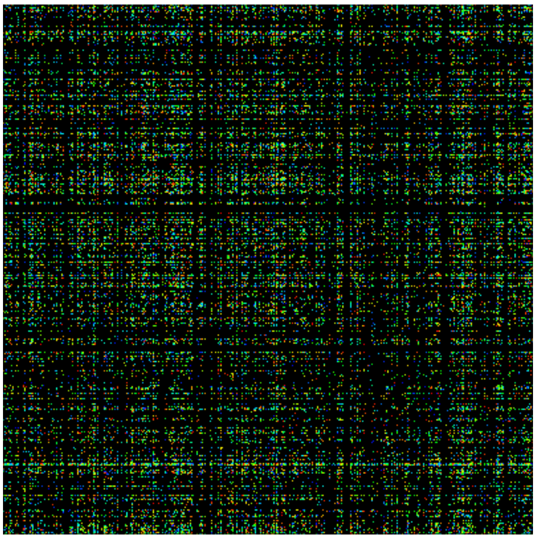


Figure 3.10: Example of Supply/Demand Graph adjacency matrix at 2006-02-01 01:00:00

As before we have calculated the probability of indirect interaction by analyzing the supply/demand graphs of every hour. The chart in Figure 3.11 depicts this probability per each cycle length. A difference with our previous measurement is that have limited the length of the cycles that we are looking for to 3 because of the larger complexity of the problem. The effect of keeping track of more swarms for the purpose of enabling indirect interaction can be directly seen from the results as there is significant increase in the probability of forming cycles of length 2 which averages to more than 0.8 and the probability of forming cycles of length 3 being almost equal to 1.

Similar to the small sample we provide some statistics on the peers as well. Figure 3.12 gives the total number of online peers, the number of multi-swarming peers and the number of lonely peers as before. As the statistics chart suggest, the consideration of more swarms results in a significant decrease in the ratio of lonely peers to multi-swarming peers. This is because tracking more swarms creates new possibilities for peer that were previously not considered to be multi-swarming to form cycles of length 2. For the same reason we also see a significant increase in the ratio of multi-swarming peers to total online peers.

Clearly Figure 3.11 demonstrates that indirect interaction between multi-swarming peers is quite probable when more information is available to study the supply/demand structure between the peers.

3.4.3 Further Analysis of Measurement Results

Our probability measurements on the FileList.org data set reveal some previously unobserved properties of the BitTorrent protocol. As both categories of measurement demonstrate the probability of indirect interaction between multi-swarming peers turn out to be

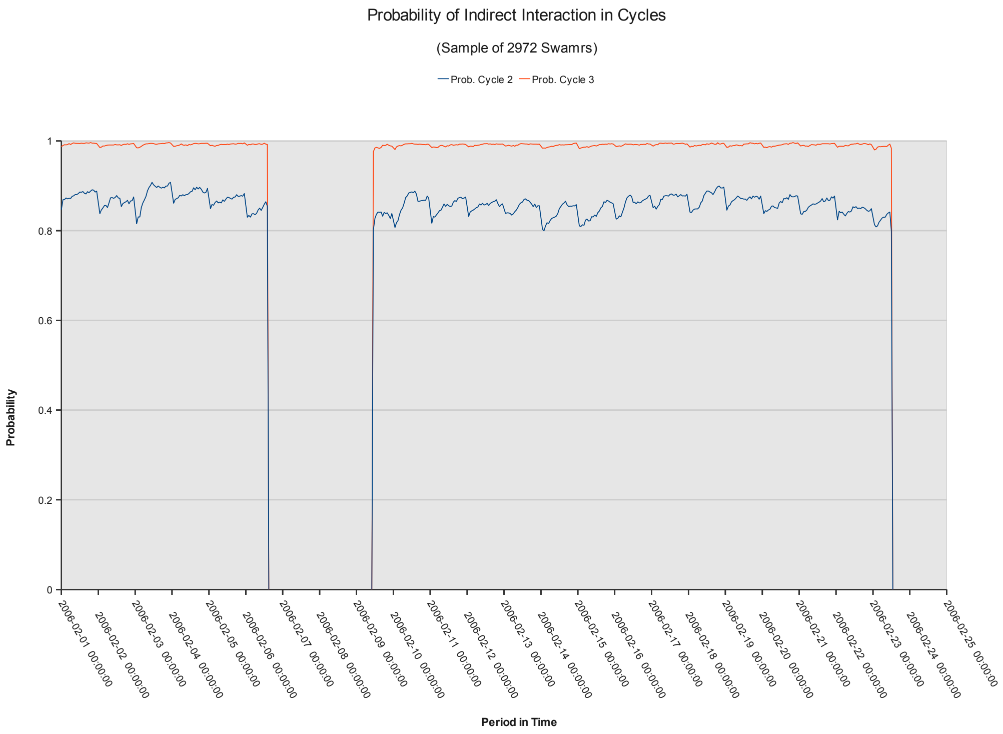


Figure 3.11: Probability of multi-swarming peers forming cycles of length 2 to 3 for 2972 Swarms.

quite significant even for the case when we consider a random sample of swarms. This is despite the observation that for our random sample of 101 swarms, the ratio of multi-swarming peers to the total number of online peers. This demonstrates that a large number of multi-swarming peers is not required to achieve high probability of indirect interaction. We expect to have a larger probability with larger numbers of multi-swarming peers though. Notice that for the larger measurement study the ratio of lonely peers to multi-swarming peers has dropped while the probability of indirect interaction has significantly increased. This indicates that a low number of lonely peers is a good indicator for high probability of indirect interaction. Moreover, it can be seen that low probabilities of cycles coincide with the periods in which a larger ratio of lonely peers to multi-swarming peers has been observed. This observation can be directly demonstrated by correlating the number of lonely peers with the probability of cycles using the *Pearson* or *Spearman* correlation method.
While the high probability of indirect interaction that we have demonstrated here might be related to the fact that our data set is from a private community of BitTorrent peers where ratio enforcement has been applied, we believe that this property of the data would not greatly impact the results. Recall from Chapter 2 that private BitTorrent communities demonstrate a high tendency to seed content. Nevertheless, we can argue why such properties should not

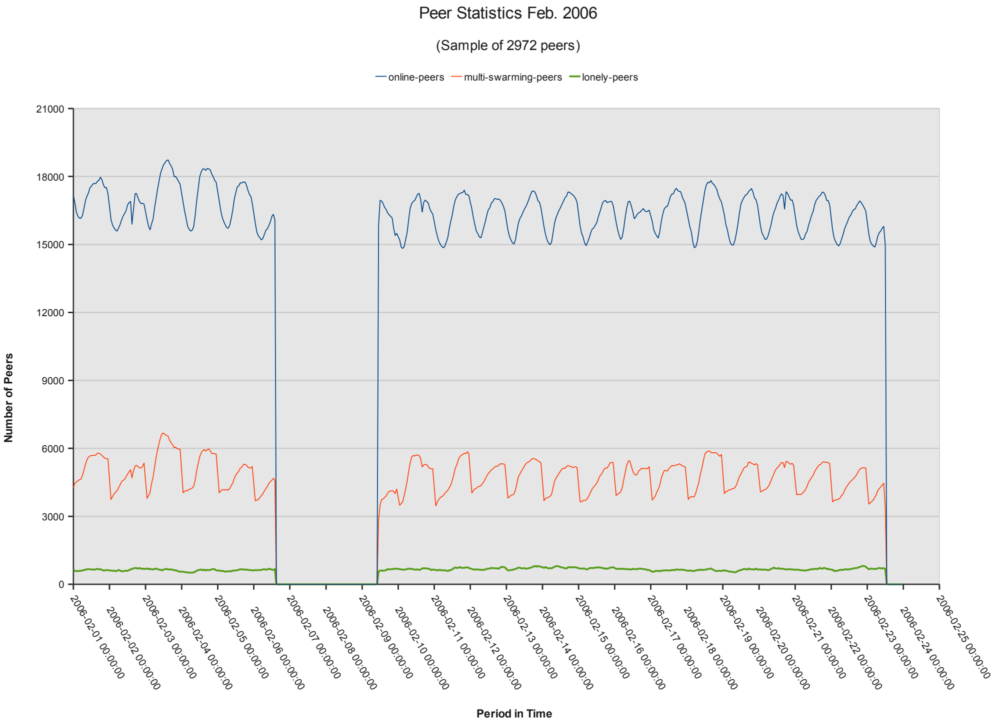


Figure 3.12: Statistics on the peers of the Large 2972 swarm sample.

have a great impact on the results that have been observed. This is firstly due to the relatively soft policy that has been applied to the FileList community. It turns out that the ratio enforcement policy of this community only limits access to newly created swarms while older swarms are still available to peers with a low sharing ratio. Furthermore, a 'good' sharing ratio that would allow full access, turns out to be smaller than a sharing ratio of 1. In comparison with more recent private BitTorrent communities like TV-Torrents where sharing ratios below 1 are not tolerated[7], the policies of FileList are quite soft and therefore should not have a significant impact on the results.[8]

A second argument relates to the definition of multi-swarming. While it is common to observe peers that seed multiple files without simultaneously leeching as the result of sharing ratio enforcement, our definition of multi-swarming does not include such behavior. Therefore, such types of peers do not appear in our results and reduce the impact of sharing ratio enforcement on our results.

[7] Such policies lead to effects that are similar to a credit crunch effect in the real world where only a small number of highly resourced peers can maintain required ratios while other less resourced peers struggle to maintain an acceptable ratio [24].

[8] We refer the reader to the work of Roozenburg et al. [46] and the Tribler website [1] for more detailed information on the FileList.org data set.

Our third argument is derived from other measurement studies on public BitTorrent trackers that show similar statistics as we have observed in out measurement study. We can demonstrate that our statistics are similar as one similar study of 2 public BitTorrent trackers [22] demonstrates that 85% of peers in the two public trackers participate in multiple swarms, which is even higher than the amount of multi-swarming peers that we have observed in our data set. Notice that 'participation in multiple swarms' is not equivalent to our definition of multi-swarming. However, a usual observation on public trackers is that swarms do not have a high number of seeders in comparison to private trackers. Therefore, we can speculate that most of the 85% of peers that have been observed in *Guo et al.*'s study are either leeching multiple files or seed only a few of they files. This means that most them should satisfy of our definition of multi-swarming. Even though our arguments support our belief about the soft effect of sharing ratio enforcement on our results, there are also more practical reasons behind our choice of the data set. All in all it seems that indirect reciprocity through indirect interaction is feasible in BitTorrent swarms.

3.5 Difficulties and The Choice of FileList

As we explained before, detailed information about every peer's online activity is required in order to study the multi-swarming behavior of peers in BitTorrent swarms. We also explained that in the case of our data set, these activities are bound to every peer's unique username, which has been assigned to it by the private tracker. However, not all BitTorrent peers are guaranteed to have unique IDs when they join multiple swarms. This is especially true when considering that some of the most large BitTorrent communities today, are formed around public trackers like the PirateBay that have millions of peers who are not necessarily uniquely identifiable. This lack of identity has been the source of many problems in general P2P systems and have resulted also made it difficult to do conduct exact measurement studies like the one that we have presented. It turns out that peers are usually assigned IDs based on some function of their IP address. We know that at least during each session of download a peer's IP address is unlikely to change. Therefore the combination of ID and IP should still allow us to identify peers during each session. Therefore such identifiers could still be used as relatively accurate measures of identifying peers in similar measurement studies. On the other hand some BitTorrent clients such as Tribler are beginning to use *permID*s that are assigned to each peer and are guaranteed to remain permanent. Such IDs can be an alternative to the combination of ID and IP addresses that could enable the accurate tracking of multi-swarming behavior. However the uniqueness and permanence of these IDs have to be reflected in the tracker activity logs.

One of the difficulties that many online mechanisms face relates to the use of *firewalls* and *Network Address Translation* (NAT). Such practices can make it difficult for peers to actually use the extended protocol that we gave as an example in the beginning of this chapter. This is because incorrectly configured firewalls or NAT devices could lead to peers not being able to contact others that are behind such devices. However, this simply comes as a natural limitation of the way the Internet operates and future work in other fields would have to address such problems.

A third factor that contributes to the difficulty of studying multi-swarming behavior is the legal issues that surround the use of P2P file sharing applications. Considering that much of the content that is distributed through these application is considered to be illegal, it is very difficult to gather information on peer activity as many trackers anonymize their logs. Privacy issues are also a contributing factor to this problem. As a result, data sets for measuring multi-swarming activity are difficult to find.

A fourth difficulty relates to the granularity of the data that can be found in logs. Usually logs contain information on peer activity that is often apart in the order of several minutes. For example our data set contains information on peer activity with intervals ranging from 5 to 10 minutes. Naturally measurements such as ours have to make assumptions that peers do not disconnect and reconnect during this interval. Nevertheless it would be unlikely to have such occurrences in our data set since it belongs to a private community and as indicated before peers have been observed to remain online for long periods in such communities.

Finally our measurement study does not consider that each of the peers that we have considered have different bandwidths. As result the cycles that we have observed in the supply/demand graphs are not necessarily high bandwidth cycles. That is the bandwidth of these cycles could be bounded by the bandwidth of the slowest peer in the loop. Naturally similar measurements in the future could also take this property into account and provide statistics on how well connected indirect interaction cycles could be. Also note that indirect interaction in cycles is only of interest while all parties have some interesting commodity to offer to the next party in the loop. Even though we have argued that interest in indirect interaction should be sufficiently long as a result of the piece distribution policies of BitTorrent, longer cycles become less and less durable. Future measurements should also take this fact into account.

Having presented our measurement results on the probability of indirect interaction and the potential of an extended BitTorrent protocol which is capable of allowing indirect interaction between peers, we dedicate out next chapter to the design of such a protocol.

Chapter 4

Mechanisms for Indirect Interaction in BitTorrent

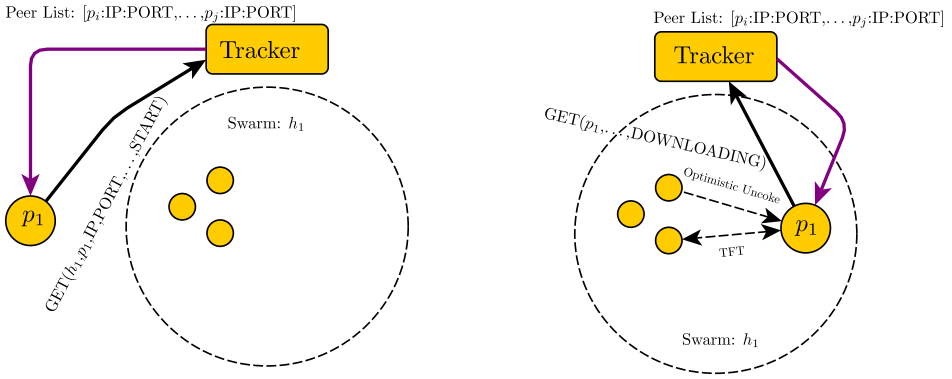
In chapter 3 we studied the potential effect of a protocol that enables the indirect interaction of BitTorrent peers over multiple swarms. That is, instead of bartering pieces of the same file, multi-swarming peers (see Definition 3.3.3) could barter pieces of different files and interact indirectly (see Definition 3.3.2). Our measurement results clearly indicate that the probability of indirect interaction, which is intuitively regarded to be low, is actually counter intuitively high even when considering a limited number of swarms and multi-swarming peers. In this chapter we discuss the detailed design of a protocol that enables indirect interaction. We refer to this mechanism as the indirect interaction mechanism (IIM) and consider the impact of this mechanism on the memory usage and workload of the tracker uses the tracker to facilitate indirect interaction. Such complexity analysis of the mechanism is essential more workload on the tracker would impact scalability and performance.

This chapter is organized as follows. We first give a detailed summary of the BitTorrent protocol in Section 4.1. In Section 4.2 we present IIM for a limited scenario of a single tracker. We present our algorithm for cycle detection and consider the impact of this algorithm on the workload of the tracker. Subsequently, we extend our mechanism for a more general case of multiple trackers in Section 4.3. These trackers communicate in order to coordinate indirect interactions among their peers and find cycles. In Section 4.4 We contemplate on the vulnerabilities of IIM and privacy that important aspects of designing distributed mechanisms. Finally we consider future possible improvements to the indirect interaction mechanism in Section 4.5 and present possible methods for decentralizing IIM to reduce dependency on the tracker and provide more scalability.

4.1 A summary of Standard BitTorrent

Before we dive into any details about the mechanisms that we are going to introduce in this chapter, detailed knowledge of the BitTorrent protocol is required so that we can explain and compare the mechanism with the standard BitTorrent protocol. Therefore, we will fist give a short summary of the standard BitTorrent protocol in this section.

A BitTorrent download starts with a peer locating a *.torrent* file of the content that it wishes to download. The initial step for every BitTorrent peer to start downloading content from a swarm is *joining* that swarm. According to the standard BitTorrent protocol a peer sends a *GET* message to the tracker responsible for the content to do so. The responsible tracker is specified in the *.torrent* file. The protocol specification states that along with a GET message, the *info hash* of the file, the peer's *ID*, *IP* address, *port number*, its *status* (an *event* of the form *start*, *downloading*, *finished*) and some information about the amount of download and upload are sent to the tracker [9]. For private trackers the peer ID is the *username* that has been assigned to the peer in the private community. For example consider a peer p_1 that wants to join the swarm h_1 where h_1 is the info hash of the file that it wants to download (See Figure 4.1(a)). To join the swarm p_1 will send a GET message to the tracker as depicted in Figure 4.1(a). In response to a GET message the tracker will reply with a list of peers $[p_i : IP : PORT, \ldots, p_j : IP : PORT]$, typically including 50 peers and p_1 will attempt to connect to each of these peers and wait for the first *optimistic unchoke* by one of these peers to receive a piece (chunk) of the file. The peer can use this piece to barter for other pieces. This process is referred to as *bootstrapping*. Once, p_1 has been bootstrapped it can barter with other peers to get new pieces of the file following a tit-for-tat mechanism. Furthermore, p_1 can regularly send GET messages to the tracker to inform it of its status and receive other peers to barter with (see Figure 4.1(b)).


Peer List: [p_i:IP:PORT,...,p_j:IP:PORT]
Tracker
GET(h_1,p_1,IP,PORT,...,START)
p_1
Swarm: h_1
Peer List: [p_i:IP:PORT,...,p_j:IP:PORT]
Tracker
GET(p_1,...,DOWNLOADING)
Optimistic Uncoke
p_1
TFT
Swarm: h_1

(a) Joining the Swarm.

(b) Bootstrapping and Bartering.

Figure 4.1: Downloading in BitTorrent.

Typically there is no control over the number of GET messages that each peer can send to a tracker, however many implemented BitTorrent clients limit the frequency of GET messages to 1 per every 30 minutes.[1] This is to keep the workload and communication bandwidth of the tracker to a minimum. However, peers can manually send GET messages with higher frequency even though it is considered to be malpractice. The tracker's peer list response can be either in the format described before or a compact format in which the peer IDs are omitted.

[1] Other implementations limit this to 20 or even 10 minutes.

Once a peer has been bootstrapped, the exchange of file chunks with other peers is negotiated through symmetrical TCP connections between peers. Peers exchange *bit strings* representing the chunks of the file that they currently posses in order to keep their neighbors up to date about what they can provide to them. With this information, each peer can request chunks from its neighbors by sending GET messages along with the index of the chunk that it seeks. This index is derived from the bit string. Peers that are interested in a chunk inform their neighbors by setting the *interested* bit of the connection to 1 and show their lack of interest by setting this field to 0. Interested peers can receive a piece if the sending side *unchokes* the connection otherwise they would have to wait for the connection to be unchoked, or alternatively, receive the chunk from another neighboring peer. A typical choice for every peer is to unchoke only 4 of its interested neighbors and optimistically unchoke 1 of its other neighbors by dividing its upload bandwidth equally among these 5 neighboring peers. As mentioned earlier in chapter 2 these are referred to as unchoke slots.[2] We refer the reader to [9] for more detailed information on the BitTorrent protocol specification and continue with presenting the indirect interaction mechanism (IIM) for a single tracker to facilitate indirect interaction.

4.2 IIM for a Single Tracker

As we discussed earlier in chapter 3, closed loop indirect interactions between BitTorrent peers facilitate indirect interaction between peers and can be created by analyzing the *supply/demand* graph between swarms. The two necessary ingredients of the supply/demand graph are information on peers that are online in each swarm and a peer identification method. Since most of the information required to build such a graph is already available at a BitTorrent tracker, it is natural to choose the tracker as a point to build and store the supply/demand graph. Such a choice would mean that a central entity in the BitTorrent protocol (the tracker) would be responsible for building and maintaining the supply/demand graph of swarms. Therefore IIM is referred to as a *centralized* mechanism.

For the case of private trackers, peers have already been assigned unique *usernames* which can be used as their identification method. However, for public trackers peers can choose their IDs and therefore the only semi-unique piece of identification information is the combination of username, IP address and port number of each peer. Since a peer's IP address and port number are unlikely to change during each session this problem will not affect the identification of peers during each session. On the other hand, indirect interactions require the peers to be connectible in the sense that if they reside behind a Firewall or NAT device, it is correctly configured to make them contactable. The mechanism would naturally only work for peers that meet these criteria.

In what follows we discuss the design of a centralized protocol for both private and public trackers with the knowledge that for the second case, peers are only *semi-identifiable* and we assume that peers have a correctly configured firewall or NAT device.

[2] The number of unchoke slots also depends on the upload bandwidth of a peer where peers with higher bandwidth will unchoke 6 or 7 connections including the optimistic unchoke slot.

4.2.1 Mechanism for a single Tracker

From our measurements in chapter 3 we already know that in the case of private trackers closed loop interactions for lengths 2 and 3 are quite probable between peers. Therefore we can assume that a limited length on the cycles of the supply/demand graph is enough to facilitate indirect interaction with high probability. Furthermore, we also know that keeping track of multi-swarming for only a limited number of peers and swarms is also enough to facilitate indirect interaction. Optimal or practical choices for the cycle length and the number of swarms that are considered depend on the circumstances and we leave these choices to be determined by future research. For now, let us assume that the tracker will keep track of all of its n swarms to create a supply/demand graph and will look for cycles of length up to k between peers. We will now describe how the private tracker can coordinate indirect Interaction between multi-swarming peers.

Informing the Tracker about Multi-Swarming

It is a heavy task for a tracker to create and keep a supply/demand graph (see Section 3.3) up to date at all times solely by itself. However, finding cycles in the supply/demand graph does not require all the information that is captured by this graph. That is, knowledge about the mere existence of an edge between any two nodes in the graph is sufficient to find cycles. To be more precise, the actual label sets of each edge of the supply/demand graph can be determined later on from the information available at the tracker. Therefore, instead of having the tracker build and maintain the supply/demand graph we can make the peers inform the tracker about the existence of an edge and gradually build a supply/demand graph that is almost up to date with respect to the actual underlying state of the peers and swarms. Removal of out dated edges occurs at regular intervals at which the trackers checks if peers have gone offline.

In order to achieve this, we propose the addition of a new status, *multi-swarming*. This status is to be used when informing the tracker of the existence of edges in the supply/demand graph. Furthermore, it can be used in the same structure as the standard GET message that peers send to the tracker to request a peer list. For example consider the peer p_1 from Figure 4.1 and suppose that it is also online in a second swarm h_2 (See Figure 4.2). In order to inform the tracker of the existence of an edge between h_1 and h_2 in the supply/demand graph, this peer will send a GET message to the tracker with the status (event) field set to 'MULTISWARM' followed by a list of info hashes of the swarms that it is a member of.

Once the tracker receives this message it can verify that p_1 is indeed a member of all the claimed swarms and update the supply/demand graph accordingly. The tracker can then use this information from the graph to facilitate indirect interaction by finding cycles. In order to do so, it will reply to p_1's GET message with a list of peers in addition to some peers with which p_1 can form a cycle. The tracker will use a portion of the message space in its reply list to accommodate the cycles that are sent to p_1 (See Figure 4.3). For each cycle that p_1 receives it will attempt to form a cycle in the direction of increasing position index in the message. For example, Figure 4.3 demonstrates a cycle of the form $p_1 \to p_2 \to p_3 \to p_1$ in

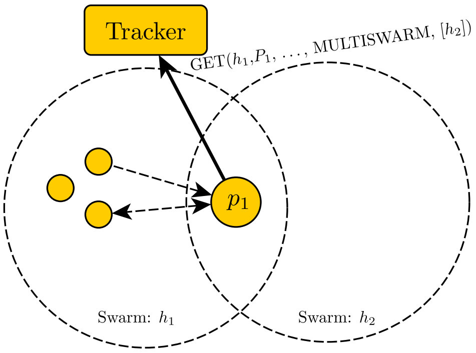


Figure 4.2: Informing the tracker of Multi-Swarming.

which p_1 would have to provide content from h_2 to p_2 who in return has to provide content from h_3 to p_3 who in return will provide content from h_1 to p_1.

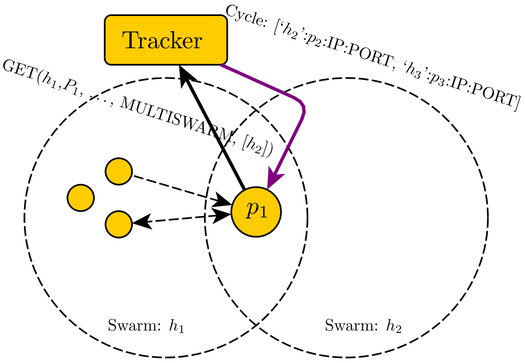

Tracker
Cycle: ['h_2':p_2:IP:PORT, 'h_3':p_3:IP:PORT]
GET(h_1,P_1, . . . , MULTISWARM, [h_2])
p_1
Swarm: h_1
Swarm: h_2

Figure 4.3: Tracker Replies with Cycles.

For every cycle that a peer like p_1 receives it has to establish a cycle accordingly with the other corresponding peers. Notice that this might bring new peers and new swarms into the picture (see Figure 4.4). The additional information that p_1 receives is actually information that it could have received through the standard BitTorrent protocol. However, a piece of information that is extra to what p_1 could have obtained in standard BitTorrent is the information regarding the activity of p_2 and p_3 in h_2. In this sense there is privacy impact to be considered and the issue should be handled with care.

After receiving the cycle from the tracker, p_1 is now able to negotiate the creation of cycles with the other peers. Multi-swarming GET messages are intended to be sent with the same frequency as standard GET messages to the tracker.

Interacting with Peers to Form Cycles

Once a peer knows how to form a cycle, it will attempt to establish connections with the corresponding peers.[3] It will do so by forwarding the cycle that it received from the tracker to the peers that are involved in the cycle. For example, p_1 in Figure 4.4 that has received a cycle of length 3 will inform p_2 of the cycle; in turn p_2 will inform p_3 and finally p_3 will inform p_1. At each stage of forwarding the cycle information, the forwarding peer is telling the next peer in the chain that it will provide content to the next peer if in return, the next peer is willing to provide content from the specified next swarm to the specified third peer which in return will do the same for the forwarding peer and so forth. The cycle ends with the originating peer receiving back it's message, which is now in a transformed state. This is because at each step of forwarding the cycle, the forwarding peer changes the messages to reflect the described interpretation.

Note that the connections that are established in the cycle are asymmetrical and once the cycle has been established data can flow in the direction of the cycle with the same tit-for-tat mechanism of the standard BitTorrent protocol. That is, peer p_1 will unchoke p_2 in return for receiving data from p_3, p_2 will unchoke p_3 in return for data from p_1 and p_3 will unchoke p_1 in return for data from p_2. This is equivalent to p_1 viewing p_2 and p_3 as a single entity with which it interacts directly.

In order to exchange pieces along the cycle we propose that at least one unchoke slot be put aside for the specific purpose of indirect interaction which is to be used as a regular unchoke slot when no cycle is available.

Having described the interaction of peers we next move on to explaining what actually happens inside the tracker to find cycles.

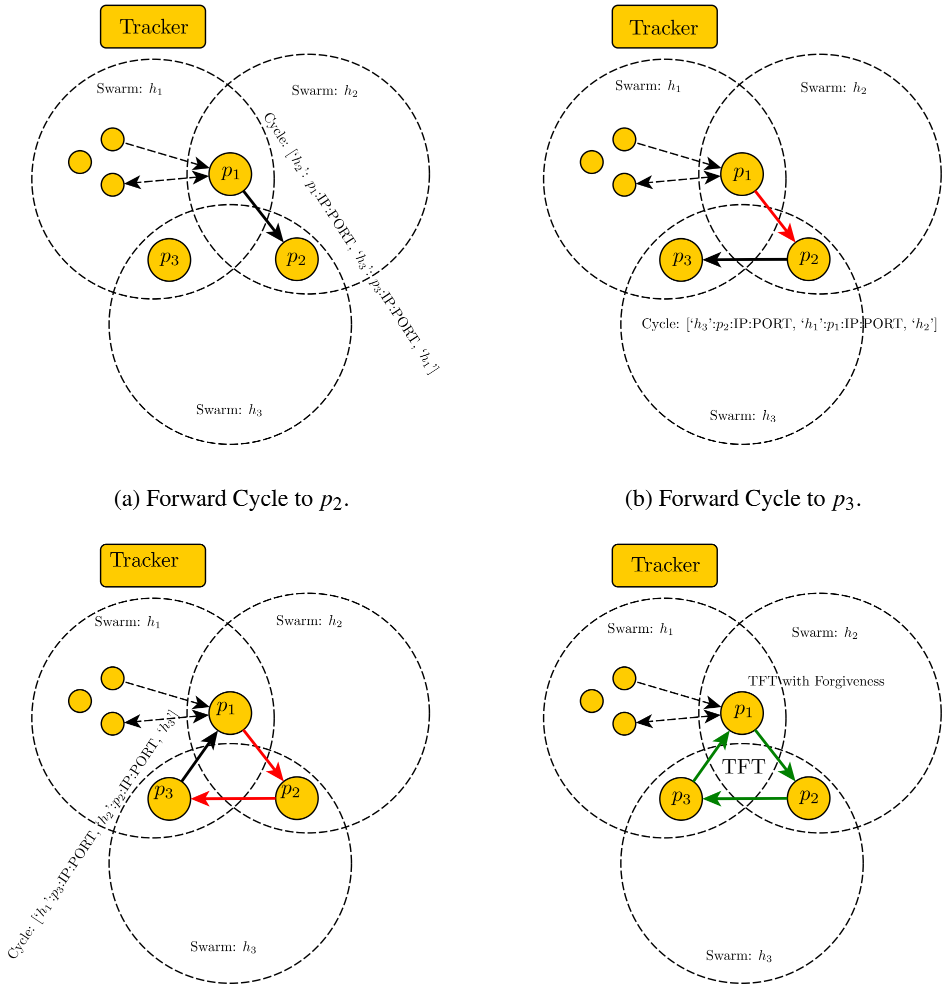
Storing the Supply/Demand Graph

As we explained before, the process of building and maintaining a supply/demand graph in the tracker is a gradual process. With peers informing the tracker about their multi-swarming activity, this graph can be built gradually and later used to find cycles. However, the way this graph is stored and the algorithm used to find cycles in the graph are yet to be explained and are very important since they can greatly impact the scalability of the mechanism.

In order to store the supply/demand graph, the tracker could use a binary version of the adjacency matrix of this graph. In this adjacency matrix a value of 1 in the ith row and j column of the matrix would represent an edge from the swarm represented by i to the swarm represented by j.

Considering that the tracker is aware of the existence of n swarms, this matrix would require n^2 bits of space to store. Note that the graph is a *directional graph* and therefore this matrix is not symmetrical. As an example consider the FileList tracker that we studied before (see Chapter 3) where n is approximately equal to 3000. This would amount to approximately 1MB of additional storage space which can easily be kept in memory. The amount of space required to store the graph is polynomial in the number of swarms.

[3] Recall that we have assumed that peers are able to contact each other.


Tracker
Swarm: h_1
Swarm: h_2
Swarm: h_3
p_1
p_2
p_3
Cycle: ['h_2':p_1:IP:PORT, 'h_3':p_3:IP:PORT, 'h_1']
Cycle: ['h_3':p_2:IP:PORT, 'h_1':p_1:IP:PORT, 'h_2']
Cycle: ['h_1':p_3:IP:PORT, 'h_2':p_2:IP:PORT, 'h_3']
TFT with Forgiveness
TFT

(a) Forward Cycle to p_2.

(b) Forward Cycle to p_3.

(c) Forward Cycle to p_1.

(d) Data Flow in the Cycle with TFT.

Figure 4.4: Cycle Establishment and Data Flow along Cycle .

Even though the space complexity of storing the supply/demand graph is polynomial in the number of swarms public trackers such as '*The Pirate Bay*' could be tracking millions of swarms[4]. If a tracker such as *thepiratebay* were to track all of its swarms in the supply/demand graph, the adjacency matrix would amount to approximately 566GBs of data, which is clearly not feasible.

Obviously, the feasible solution for such popular trackers is to create and maintain the matrix for only a limited number of swarms. We have already demonstrated that this is a valid option by our measurements in Chapter 3. A policy of keeping track of a combina-

[4] On September 22, 2010, *thePirateBay* indicated 2,204,802 swarms on its homepage: `http://thepiratebay.org`.

tion of popular swarms and newly created swarms could also prove to be a good solution, however, we will leave the determination of such policies for future work.

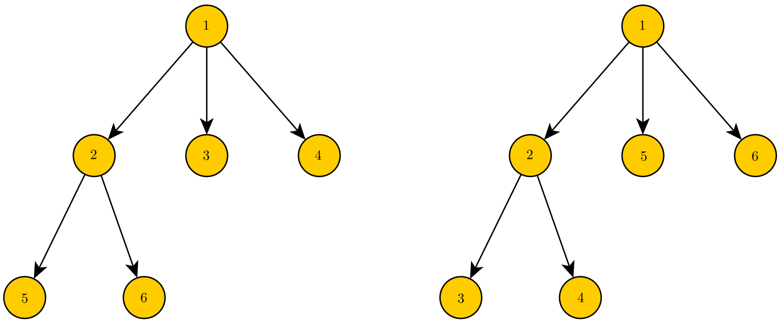
An alternative to keeping the adjacency matrix of the supply/demand graph is to store an adjacency list of the graph. In this representation a list of adjacent nodes is stored for each node in the graph. However, this representation can grow much larger than the adjacency matrix if the supply/demand graph contains many edges. Furthermore, the lists would have to store integer values instead of bits. The advantage of an adjacency list however, is that finding the adjacent nodes of the graph to a given node can be accomplished in constant time while for the adjacency matrix this requires $O(n)$ time in worst case. Since, our measurements of chapter 3 show that there are a large number of edges in the supply/demand graph, it seems more appropriate to use the adjacency matrix.

On a separate note, we would like to keep the adjacency matrix of the graph in memory for performance purposes. Hence, choosing a limit on the number of swarms from which the supply/demand graph is generated, depends on the amount of memory that is available. For example with 16GBs of available memory, we could store approximately up to 370,000 swarms. Nevertheless, for the sake of keeping to our previous assumptions let us still assume that the tracker will generate and maintain the supply/demand graph of all of its n swarms and continue with describing how the tracker could use this matrix to find cycles.

Finding Cycles in the Supply/Demand Graph

Given the adjacency matrix of the supply/demand graph, there are several *polynomial time* algorithms to find cycles within the graph. Recall that we have limited the length of the cycle that we are looking for to a constant k. Therefore, we can use a depth $k-1$ limited exploration of all possible paths in the graph given a starting point. This approach is naive when seeking cycles of length 3 and more but since cycles of length 2 are quite probable we consider this approach as a valid candidate. With a length of 2, a breadth first traversal (see Figure 4.5a) of all possible paths in the graph finds all possible cycles with a time complexity in the order of $O(n)$ (linear scalability). For cycles of greater length the naive approach has a complexity in the order $O(n^{k-1})$ since each node has n neighbors in worst case and n^{k-1} nodes of the graph have to visited. Clearly the naive approach does not scale well if cycles of length more than 2 are sought.

An alternative to the naive approach of traversing all possible paths is to use the well known Breadth First Search (BFS) or Depth First Search (DFS) algorithms. Both BFS and DFS construct a search tree from the graph and find cycles with a time complexity in the order of $O(\|E| + \|V\|)$ where V is the set of nodes in the graph and E is the set of edges in the graph. The difference is in the order in which they traverse the graph to find cycles (see Figure 4.5). One disadvantage of the BFS and DFS algorithms is that even though they guarantee to find a cycle if one exists the cycle might not be valid to use for indirect interaction. A simple example is when a peer is leeching in two swarms. By convention, this creates a cycle between the two swarms in the supply/demand graph (see conventions in Section 3.3). Nevertheless this cycles is not feasible because only one peer appears on the label sets of the edges and a peer cannot indirectly interact with itself. On the other hand the naive approach guarantees that it will find a valid cycle if one exists but it comes at a greater time complexity and hence less. Notice that for cycles of length 2 the naive approach to finding cycles from before is essentially the same as BFS.



(a) Breadth First Traversal of a Graph. (b) Depth First Traversal of a Graph.

Figure 4.5: Traversal of a Graph.

Once a cycle has been found by either of the algorithms, the tracker would have to compute the label sets of the edges along the cycle[5]. If the ith swarm in the cycle contains m_i peers and the next swarm along the cycle contains m_{i+1} peers, computing the label sets of the edge can be accomplished by computing the intersection of the i and $i+1$th swarms with time complexity $2 \times O(m_i + m_{i+1})$ (provided that the peers for each swarm are sorted). Since the cycle lengths are limited to k, computing the label sets would have to repeat $k-1$ times in worst case. Therefore the complexity of finding valid cycles from a single cycle in the supply/demand graph is $k \times O(max_i\{m_i\})$ in worst case. As a result, the worst case scenario of finding cycles between peers is polynomial in n, k and the maximum size of a swarm M. In the case of the naive approach this amounts to $k \times M \times O(n^{k-1})$ and for the BFS/DFS algorithms to $k \times M \times O(\|E\| + \|V\|)$.

Since our measurements from Chapter 3 show that cycles of length 2 are significantly probable we choose to limit cycle lengths to 2 from this point on and use the naive algorithm for finding cycles.

In order to demonstrate the impact of this algorithm on the scalability of IIM we have implemented a tracker that we have given real data from the FileList.org measurements from before. We have created a scenario in which an increasing number of peers simultaneously request cycles of length 2 from the tracker and have measured the average time that it takes for the tracker to respond to these requests. In this scenario there are a maximum number of 4463 multi-swarming that make simultaneous requests from the tracker. Figure 4.6 depicts our measurement of the response time. The linear impact of the cycle finding algorithm is evident in Figure 4.6 and it demonstrates that IIM scales linearly.

[5] Recall that the supply/demand graph only indicates the existence of an edge.

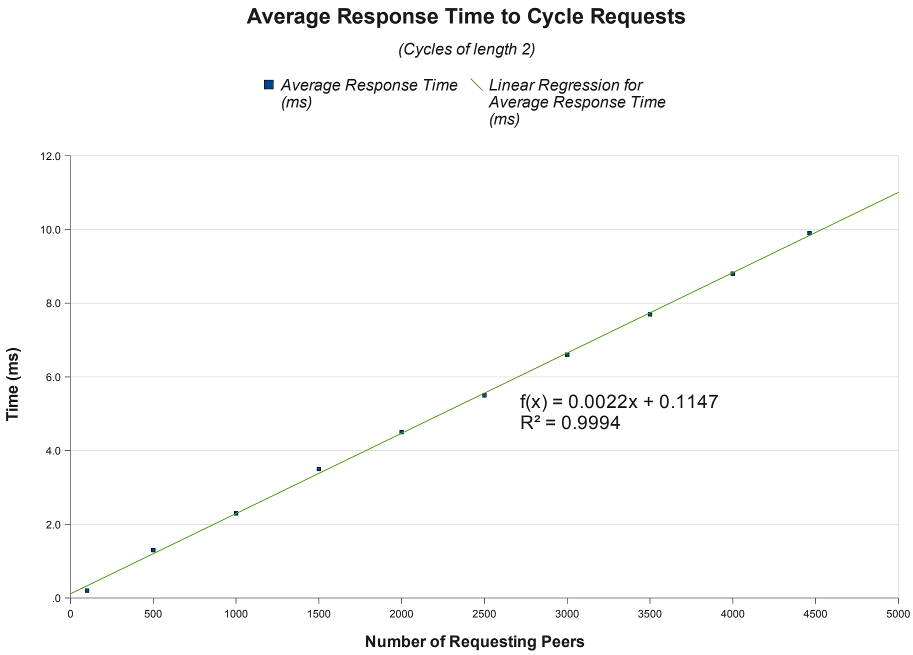


Figure 4.6: Scalability of the indirect interaction mechanism.

Updating the Supply/Demand Graph

In order to keep the supply/demand graph up to date the tracker would have to simply removes edges that no longer exist. Removal of edges can be triggered in two ways. First a peer might inform the tracker that it wishes to disconnect from a swarm. Second a tracker performs a periodic check of a peer's status and discovers that it has not received a status update from that peer for a certain amount of time.

If a peer sends a disconnect message to the tracker the tracker firstly checks if the peer has been multi-swarming. If not the supply/demand graph does not require to be updated. If otherwise, the tracker will intersect the set of peers in the swarm from which the peer left with the set of peers of all the other swarms in which the peer is still online. If any of the intersections turns out to be empty edges between the two intersected swarms have to be removed from the supply/demand graph. This can be accomplished with a time complexity of $O(M)$ where M is the maximum number of peers in the swarms.

Similarly if the tracker discovers that it has not received a status update message from a peer it will follow the same procedure as before and update the graph accordingly.

Clearly the major impacting factor in the scalability of IIM is the cycle finding algorithm as the complexity of updating the graph is less than the complexity of finding cycles.

At this point we have described all the necessary details required to implement IIM for a single tracker. In our next section we extend IIM for use with multiple trackers.

4.3 IIM for Multiple Trackers

A similar supply and demand approach can be used to coordinate indirect interactions between peers of multiple trackers. The idea is for trackers to use a similar supply/demand graph in which the nodes of the graph represent trackers instead of swarms. An edge in the supply/demand graph of trackers from tracker i to tracker j means that some peer is leeching in a swarm of tracker i and leeching/seeding in a swarm of tracker j. A difference with the supply/demand graph from before is that information regarding the peers that appear on the label set of the edge are distributed among the two trackers i and j. Therefore, to coordinate the interaction of such peers trackers i and j need to exchange information to find such shared peers. One way to do so it to for trackers to form an overlay network in which they can communicate.

4.3.1 Tracker Overlay

To coordinate the creation of cycles between peers that are online in multiple trackers, trackers can form an overlay network in which they can exchange information regarding their multi-swarming peers. For example consider the overlay network depicted in Figure 4.7. This overlay network for example allows tracker 1 to communicate information regarding its multi-swarming peers with trackers 2, 3 and 5. Furthermore, notice that the overlay network of trackers need not necessarily connect all existing trackers to each other. That is, a single tracker can connect to a small selected number of tracker to facilitate information exchange.

The selection of which trackers to connect to can be based upon multiple criteria. For example trusting the organization that runs each tracker, popularity of each tracker and its workload could all be considered influencing factors. Moreover, the trackers that are connected in an overlay need not necessarily be run by multiple organizations. That is a single organization that operates multiple trackers could connect its trackers in an overlay network that is completely disconnected from the outside world.

The important issue here is that each of the trackers that are connected have a local view of their swarms and their peers. As mentioned before, information regarding the peers that are active in multiple trackers is actually distributed between the trackers therefore, no single tracker has a global view of the activity of peers.

A simple, yet naive approach to facilitating indirect interaction between the peers of connected trackers is for each tracker to keep a global view of all the existing swarms. The trackers would communicate all activity that crosses the boundary of a single tracker and into the boundary of another tracker. That is, each tracker can include the swarms of the adjacent trackers in their local supply/demand graph and communicate in order to compute the label sets of edges in the global supply/demand graph. However, the storage requirements and bandwidth consumptions of such an approach could easily exceed the capabilities of

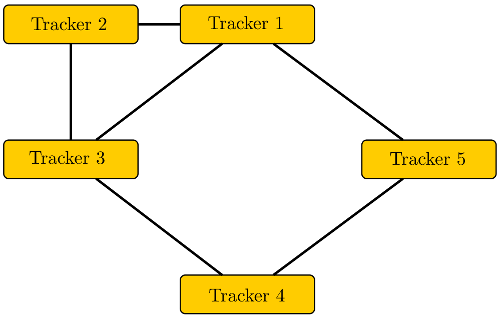


Figure 4.7: Overlay Network of Trackers.

a single tracker. As a result this naive solution is not attractive from a scalability point of view nor from a performance point of view.

Nevertheless, another simple and yet perfectly scalable approach to global tracking is possible if a global tracking mechanism limits it self to finding cycles of length 2 between peers that cross the boundary of a single tracker. Given that cycles of length 2 are quite probable for peers of a single tracker it is logical to assume that cycles of length 2 are also probable between peers of multiple trackers (if not more probable), because .torrent files usually contain multiple trackers. This means that peers are also likely to form cycles when active in multiple trackers. The following subsection describes how this simple assumption leads to a scalable mechanism for the creation of cycles of length 2 between peers of multiple trackers.

4.3.2 Mechanism for Multiple Trackers

Similar to the single tracker mechanism from Section 4.2 we need to explain the process by which the supply/demand graph is created, stored, used to find cycles and kept up to date.

The process of informing the tracker of multi-swarming behavior can be accomplished in the same way the mechanism for a single tracker operates. That is, a peer can inform the tracker of multi-swarming with the ‘multiswarm’ GET message by also including the tracker IP address in the GET message and the list of swarms that it is active in. The only difference is that there is a limit to the number of information pieces regarding activity in other trackers that each tracker is willing to store. The other operations of the mechanism however, are some what different and their details follow.

Storing the Supply/Demand Graph of Trackers

As we mentioned before, the naive approach of storing the multi-swarming information of adjacent trackers, is not scalable. However, if the mechanism is limited to finding cycles of length 2, each tracker does not require to expand its supply/demand graph to reflect every tracker that it is connected to in the overlay network. With the limitation on cycle length, it is enough for each tracker to only keep track of its incoming edges in the supply/demand graph of trackers.[6] For example consider tracker t_1 from Figure 4.8. For every peer that is active in tracker 1, a limited number of activities that create incoming edges to the tracker from adjacent trackers could be stored. The depicted peer p_2 in Figure 4.8 is a leecher in swarm $t_2 : h_1$ and a seeder in the swarm $t_1 : h_1$ which creates an incoming edge from tracker t_2 to t_1 in the supply demand graph. When informed by p_2, tracker t_1 creates a limited view of the supply/demand graph between itself and tracker 2. Similarly, the tracker t_2 creates a limited view of the supply/demand graph from the information that is has on p_1.

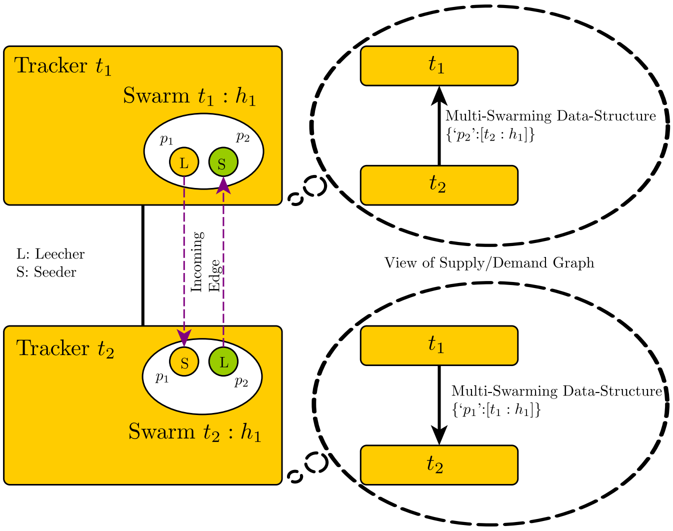


Figure 4.8: Global Tracking Supply/Demand Graph. Tracker t_1 knows that it has an outgoing edge to t_2 and vice versa.

As a result of the limited view of each tracker, trackers do not need to store an adjacency matrix of the global graph. Instead, a tracker can store a limited number of multi-swarming records per active peer. A reasonable limit is to store 4 records per peer, since the standard number of active bartering connections per peer are usually set to 4 in the BitTorrent protocol and therefore a peer will only use 4 cycles at most. Therefore, each tracker could store a data structure of the following form to have a local view of supply/demand and use it to find cycles of length 2:

$$\{p_i : [t_i : h_j, \ldots], p_{i+1} : [t_k : h_l, \ldots], \ldots\}.$$

We call this data structure the global supply/demand view (or global supply/demand data structure or global supply/demand for short). The entries in this data structure are indexed by each peer p_i followed by a list of swarms from other trackers of the form $[t_i : h_j, \ldots]$. Each list entry indicates a tracker and the swarm in which the peer is active. There are many other possible ways of keeping this information (e.g. storing this information in a

[6] The tracker could also keep track of its outgoing edges but the choice of incoming edges is more efficient in terms of storage demand and the size of the messages that will be passed around between trackers. In the case of incoming edges the types of the peers also need to be stored and passed around.

data base) for which more efficient indexing or storage requirements can be accomplished but the point here is to demonstrate how this information is used. An interesting fact about keeping a local of view of the graph is that for a tracker of a similar size to FileList this data structure amounts to only 11 MBs of additional storage space when all 110000 members are online and multi-swarming in at least 4 different trackers. The same data structure amounts to 2700 MBs if all $27,000,000$ peers of thePirateBay tracker[7] were multi-swarming in at least 4 different trackers. Clearly the first example is a feasible solution. Nevertheless, size limitations on the data structure could be adjusted according to the capabilities of a tracker.

Next we move on to explaining how the trackers could use their local view to find cycles.

Finding Cycles Between Trackers

Given the local view of the global supply/demand graph, a tracker resorts to looking up the global multi-swarming information of a peer in order to find cycles for that peer. Consider a peer q that has requested a cycle from the tracker. The tracker could find the corresponding pieces of information of the peer from the described data structure and find out the trackers that the peer is active in. Note that the tracker also knows about its own swarms in which q is active in addition to q's type. The tracker can then communicate one by one with the corresponding trackers through the overlay network with a message indicating that it is looking for a cycle for q. To elaborate consider the following example:

Example 4.3.1. *Peers q is seeding in the set of swarm $\{t_1 : h_2\}$ of tracker t_1 and is also leeching in the of swarms $\{t_2 : h_2\}$ of tracker t_2. This peer has requested a cycle from tracker t_1 and has already informed t_1 of its multi-swarming in tracker t_2. Figure 4.9 depicts this situation.*

In order to satisfy the request from q from in Example 4.3.1, t_1 will send a message to t_2 telling it that it is looking for peers that are active in the swarm $\{t_1 : h_2\}$ and are also active in the swarm $\{t_2 : h_2\}$. Subsequently, t_2 will look for peers that meet the first criteria in its global supply/demand data and then use the second criteria to come up with a list of peers that q can from cycles with. Figures 4.9 and 4.10 demonstrate this mechanism in two steps. Once t_1 receives the peer list message of t_2 it can send the cycles to q in the same way as we described before in Section 4.2. Note that all operations required to find peers that meet t_1's criteria can be accomplished in polynomial time. More Specifically this can be achieved in $O(n_2)$ time where n_2 is the number of multi-swarming peers in t_2. Furthermore, the communication between trackers which takes place in two steps is also efficient because the first step only requires the transmission of 80 bytes and the second step only requires the transmission of 104 bytes for returning at most 4 cycles of length 2. Note that the messages do not actually require to transmit the tracker ids that are depicted in the figures because each tracker knows to whom it is communicating. The tracker IDs (e.g. t_1 and t_2) in the figures are depicted for the clarity of the examples.

[7] The number of peers are taken from the piratebay home page on September 22, 2010

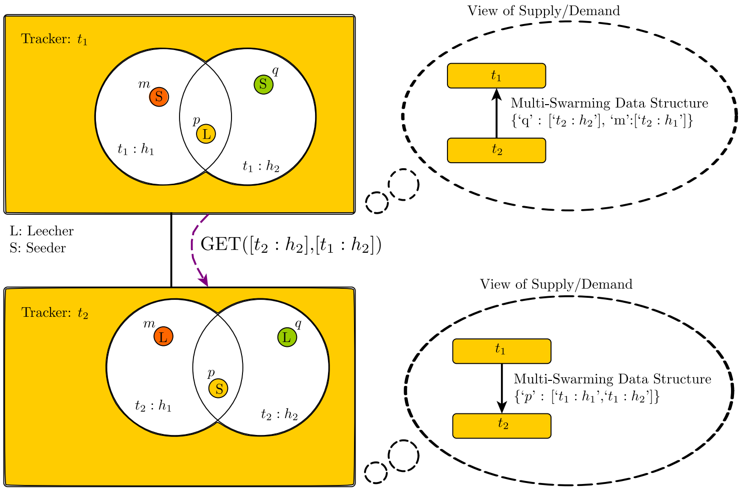


Figure 4.9: Tracker communication in the first step of cycle finding.

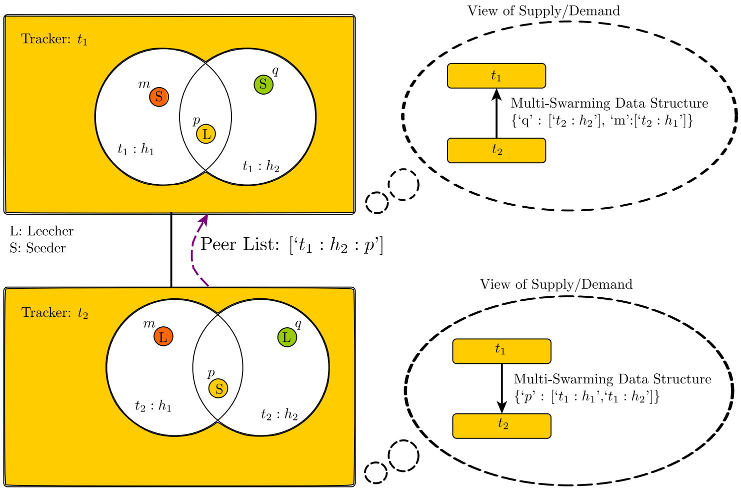


Figure 4.10: Tracker response with peers that can form cycles of length 2.

Updating the Supply/Demand Data Structure

In order to keep the global supply/demand view up to date each tracker would simply have to remove the record of a peer that has just gone offline. While peers could announce to the tracker that they have gone offline, some peers simply don't. Therefore as before the tracker regularly checks if it has received a status update messages from peers. If a peer does not send an update message within some certain period of time it is considered to have gone offline. The tracker could simply update the global supply/demand view with the same frequency at which the a standard BitTorrent trackers checks for status updates from peers.

Once a peer has been determined to have gone offline the tracker will simply remove that peer from the global supply/demand data structure. As a result, keeping the data structure up to date is no extra burden on the tracker because it can be accomplished in constant time.

Even though the mechanism that we have described here is quite scalable in terms of both storage and communication demand, there are still possible ways to improve both. The next section describes some ways of further improving the mechanism.

4.3.3 Further Improvements to the Global Mechanism

Up to now we have considered that leechers and seeders can both benefit from indirect interaction. Nevertheless, the main point of indirect interaction has been to incentivize seeding behavior which is left out of the standard BitTorrent protocol. While the TFT policy of standard BitTorrent can to a great extent incentivize leechers to provide content, the point has been that seeders do not require content unless they are leeching in some other swarm as well. Since the single tracker mechanism of Section 4.2 is quite likely to find cycles for peers (regardless of their type) with high probability, it seems that there is no point in having a global tracking mechanism. However, the global tracking mechanism could be considered as a backup plan for peers that are online in multiple swarms within the boundaries of a single tracker but are not leeching any content in the same boundary. That is, these peers are seeding content in one tracker and leeching content in another. It makes sense for a tracker to incentivize these seeders to stay and provide content in return for helping them receive content in other trackers. If the extended indirect interaction mechanism is only used in such cases storage requirements and communication frequency with other tracker is significantly reduced. Let us elaborate.

If we consider the global tracking mechanism as a backup plan, the entries in the global supply/demand data structure will be far less than the worst case scenario that we described earlier. Regular multi-swarming peers could use the single tracker mechanism to receive cycles while peers that seed in one tracker and leech in another could resort to the backup mechanism. Notice that the global helps create incentives for a larger number of peers in combination with the single tracker version.

Now that we have described our centralized mechanisms we will discuss some of the issues that are very important but some what separate from the details of the mechanism.

4.4 Issues

Two of the main issues with the use of online mechanisms (centralized or decentralized) are the issues of security and privacy. By security we mean the inability of malicious entities to exploit mechanisms to their advantage. By privacy we mean the safety of personal data or the consent of the users to make certain information available. In this section we discuss some of the main properties of the previously described indirect interaction mechanisms in terms of these issues.

4.4.1 Security

It is very important for an online mechanism to be resistant or secure to misbehavior or malicious behavior by participating entities that aim to use the mechanism in a way that was not intended. Usually these unintended ways of use are more profitable to pursue which make them more attractive choices.

From a game theoretical point of view, a mechanism would have to be *incentive compatible*, meaning that it has to benefit peers in such a way that misreporting or lying about information to the mechanism does not lead to a higher payoff for the lying peer. In this sense, both of the mechanisms that we have provided are incentive compatible for two reasons. First, the information about multi-swarming that a peer sends to the tracker can be verified because of the centralized nature of the tracker. That is, the tracker already has this information and a peer will not benefit from lying that it is online a swarm which it is not. The mechanism only requires peers to declare their activity in order to build a supply/demand graph gradually and reduce the burden of the tracker. Second, suppose that the tracker does not verify the multi-swarming information that it receives from the peers. In that case, the peer again does not benefit from lying because firstly, it has to be able to provide content to peers that it supposedly forms cycles with (which it cannot) and secondly it will receive less peers to interact with in comparison with the case in which it had not lied to the tracker. As a result the peer would have been better off by not lying and not requesting cycles from the tracker while it was not multi-swarming. Therefore, there is no incentive to trick the mechanism.

A second positive aspect of IIM is that it is sybil-proof. As explained before sybil attacks [12] involve a malicious peer impersonating multiple identities and faking transaction in order to gain some reputation. Since, IIM does not involve reputation scores, sybil attackers cannot benefit from faking transactions. Another way to look at this is to assume that indirectly interacting peers issue temporary reputation scores to each other that cannot be cumulated and transfered to be used in future interaction. In this sense the reputation scores that are issues have to be used immediately otherwise they would expire. Therefore the reputation gained from fakely interacting with a one's own sybil cannot be used in other transactions. As a result of this property IIM also tends to be more fair and prevent discrimination against less reputable peers by highly reputable peers because no reputation can be cumulated. This is in contrast with sharing ratio enforcement methods in which such discrimination is possible as highly resourced peers can gain higher reputations.

A third positive aspect of IIM is that unlike reputation systems it cannot be used to launch distributed denial of service attacks. Since peers have no control over the cycles that are introduced in response to requests for indirect interaction there is no way that a malicious peer could target a single node on the Internet and cause a surge of messages being sent to the target of the attack.

Nevertheless, the mechanism is not perfect and some ways of exploiting the mechanism are possible. For example it is possible to imagine that peers could strategize and withhold to reveal information regarding the pieces of a file that they currently posses in order to prolong the interest of other parties. That is, peers could strategically declare their possession of a piece at a moment when there are almost no more file pieces that they can provide to their neighbors in order to keep them interested for longer. In fact, Levin et al. have demonstrated such an exploit strategy for the standard BitTorrent protocol and successfully implemented a client that strategizes based on the same technique[30]. However, in the case of interacting in cycles, prolonging the interest of other peers is actually a positive strategy as long as a peers' choice of withholding information does not disrupt the data flow between the peers. It turns out that strategizing clients such as the one developed by Levin et al. are almost never encountered in practice and only appear in the confines of laboratories. Furthermore, Levin et al. have also demonstrated that it is actually infeasible to force a monopoly on the possession of a piece which could be considered as the ultimate goal of such strategies . Therefore, we can safely assume that the mechanism is unlikely to encounter suffer from such exploits.

A separate concern in online mechanisms is the spreading of pollution or incorrect information. Clearly, the verifiability of the data that is send to a tracker prevents pollution from spreading in both mechanisms. For the single tracker mechanism pollution is immediately recognized and for the global tracking mechanism the trackers would recognize polluted data as soon as they communicate to find a cycle for a misbehaving peer. Also notice that junk data and polluted file pieces cannot be exchanged in the cycles because each peer posses the hash values of the pieces to verify their integrity (either an SHA hash value or a Merkel hash value).

Finally IIM is vulnerable to an exploit known as 'the large view exploit' [50]. This exploit is a direct result of the tracker not being in strict control over how many requests a peer can make in a certain period of time. The large view exploit involves a peer sending requests to the tracker with a frequency higher than the intended 'one request message per every 30 minutes' as specified by the BitTorrent protocol. As a results a peer would improve its chances of receiving optimistic unchokes by neighboring a large number of peers and downloading a file faster.

4.4.2 Privacy

Another main concern with online mechanisms is the issue of privacy. It is essential for users to know or at least receive a notice that via the use of IIM they could be sending out information regarding their activity in BitTorrent swarms to other entities such as the tracker or other peers. To demonstrate the point that we are trying to make, recall the example from Figure 4.4 in which three peers are establishing a cycle of length 3. During the process of cycle establishment each peer receives information about the activity of a peer in a separate swarm which they would not normally be able to acquire with the standard BitTorrent protocol. For example peer p_1 finds out about the activity of p_2 and p_3 in swarm h_3 and likewise p_2 about p_1 and p_3's activity in swarm h_1 and so forth. This is considered to be a privacy issue.

Nevertheless, many BitTorrent clients have implemented decentralized ways of tracking peers [46, 39] through which each peer is able to find information of the type that we described before. Such decentralized mechanisms have been implemented as a backup for times in which a tracker is non responsive or offline. They are intended to reduce the dependency of peers on a single point of failure; the tracker. Even though the same type of information can be revealed in such decentralized mechanisms, privacy issues have not deterred users from using decentralized tracking mechanisms for reasons that are unclear to us. Another example is the popular use of DHTs in BitTorrent for peer discovery. Theoretically it is possible to acquire the same type of problematic information through the use of DHTs. However, DHTs are a popular part of many BitTorrent clients.

Having discussed some of the important aspects that distinguish IIM from current incentivizing solutions (e.g. reputation systems and sharing ratio enforcement) we move on to describing other areas that we consider to be potential improvements to the mechanisms that we have described in this chapter. We address the issue of decentralization of the BitTorrent protocol and propose some potential techniques that support the indirect interactions that we aim to create with our centralized mechanisms.

4.5 Future Work

One of the established trends with respect to the BitTorrent protocol is the decentralization of the tracking mechanism which aims at reducing or eliminating the dependency of peers on the tracker for discovering other peers. This is because the centralized nature of the BitTorrent tracker makes it a single point of failure. In fact, due to both performance issues and legal issues many BitTorrent trackers are facing difficulties.[8] As a result of such issues, three different solutions have emerged: **i**) multiple trackers for content **ii**) DHTs for decentralized tracking of peers and **iii**) epidemic protocols for decentralized tracking and content discovery [44, 46].

While the first approach can still benefit from the mechanisms that we have described in this chapter the disappearing trend of trackers indicates that the other two approaches are more likely to be useful. While theoretically the same information that is available at a tracker should also be available in a DHT, the distribution of data among peers, means that a supply/demand graph of the type that we described before cannot be built in the same way to find cycles. With a decentralized tracking mechanism it is upon each peer to find possibilities of indirect interaction by itself. Given that the DHT implementations that are currently being used by BitTorrent clients such as the Mainline implementation[9] or Vuze[10] are facing difficulties with scalability and performance [39] they do not provide good candidates for the implementation of indirect interaction. Even though it should theoretically still be possible for multi-swarming peers to find other multi-swarming peers through a DHT by continuously requesting peers from multiple swarms in which they are also active. The exact amount of requests that they have to make to find at least one other multi-swarming peer to interact with in a cycle of length 2 or greater is unknown and most probably not a scalable solution. Therefore, DHTs are not considered an attractive solution for decentralized indirect interaction mechanisms.

On the other hand the epidemic protocols for tracking and content discovery such

[8] The piratebay is one of the recent examples which had to ensure legal use of the protocol or shutdown if it fails to do so.

[9] The standard BitTorrent Client.

[10] Formerly known as Azureus.

as BuddyCast [44] that have been implemented in the Tribler [1] BitTorrent client show promising potential.

4.5.1 Decentralized IIM Based on Epidemic Protocols

Epidemic protocols are based on the idea of how epidemic diseases spread and are also referred to as gossip protocols due to their similarity with rumor spreading. Epidemic information spreading is highly scalable and robust and there is an exponential decrease in the ratio of the population that have not received an information piece after a certain amount time or rounds of information exchange[46].

BuddyCast [44] in an epidemic protocol implemented in the Tribler BitTorrent client. Tribler peers exchange information regarding the files that they have downloaded over time with peers that have a similar download history. Peers also exchange information with randomly selected peers periodically. Similarity is determined by a *similarity function* and peers that have similar download history or have similar ratings for content are considered *buddies* based on the scores that are assigned by this function.

Including each peer's current download activity in the information exchange of the epidemic protocol would result in multi-swarming peers that are downloading the same content receiving similar scores and discovering each other. For example, consider two peers p_1 and p_2 where p_1 is leeching content in the swarm identified by the SHA1 hash h_1 and seeding in the swarm h_2 while p_2 is doing the exact opposite of p_1. Both p_1 and p_2 lie in the intersection of the set of peers that are active in swarms h_1 and h_2 and by definition have a greater similarity than peers that are only active in one of the swarms. As a result, the BuddyCast information exchange between the peers of either swarm could lead to the discovery of p_1 by p_2 or vice versa and they can both infer from the information that they have received through BuddyCast that they are able to form a cycle of length 2. Therefore indirect interactions within cycles of length 2 are quite natural to an epidemic protocol such as BuddyCast and there is a potential for finding cycles without the need for a centralized tracker and the supply/demand graph concept that we used. Furthermore, using a separate scoring function that would assign higher similarity scores to peers that are in similar situations as the peers of our example (i.e. doing opposite activities) could result in an even better matching of peers that can form cycles of length 2.

On the other hand, the inference of the existence of cycles of greater lengths from the information that a single peer receives through BuddyCast should theoretically be possible as well but it would naturally require a wider spread of information in comparison with the previous case. For example consider the situation that was depicted earlier in Figure 4.4 between three peers that formed a cycle of length 3. All three peers would receive the same similarity scores because pairwise they are sharing content from shared swarms. If at least one peer has received information regarding the activity of the other two peers in the third swarm it is able to infer the existence of the cycle. However, the ability to infer the existence of this cycle should be less probable than the inference of the existence of cycles of length 2. This probability should decrease as the length of the cycle increases because there are more missing pieces of information that are required to infer cycles of greater length. Luckily though cycles of length 2 should be enough to create the seeding incentives that we aim to create and therefore epidemic protocols such as BuddyCast are considered one of the potential candidates for the facilitation of indirect interaction.

Up to this point we have theoretically described indirect interaction in BitTorrent and we have provided a detailed design for a centralized mechanism that facilitates indirect interaction in cycles. Our next chapter is dedicated to providing game theoretical models of IIM for the purpose of studying the incentive structures of the mechanism. These models demonstrate in mathematical terms the advantages of indirect interaction and prove that indirect interaction incentivizes seeding.

Chapter 5

Theoretical and Experimental Analysis of the Mechanism

In Chapter 4 we introduced an 'Indirect Interaction Mechanism' for a single BitTorrent tracker the aim of which is to incentivize peers to seed content (or alternatively provide more content) as a whole. Up to this point however, we have neither theoretically nor empirically provided any evidence of why the proposed mechanism should achieve such a goal. This chapter is dedicated to providing theoretical (Section 5.1 and5.2) evidence.

The chapter is divided into three main sections. In our first section we provide a basic game theoretical model of the BitTorrent protocol (Subsection 5.1.1) which is later extended to explain the indirect interaction mechanism (Subsection 5.1.2). Based on this model, theoretical analysis of the mechanism's properties is provided and hypotheses are derived (Subsection 5.1.3) which are later examined through simulations of the mechanism (see Chapter 6). The second section of this chapter is dedicated to an experimental game theory approach to our mechanism. We explain why such an additional approach is useful (Subsection 5.2.1). Furthermore, we report some experiments with human subjects from the literature which roughly model our mechanism. We Introduce the 'public goods game' (Subsection 5.2.2) and a variant of this game combined with the 'indirect reciprocity' game (Subsection 5.2.3) which is used in these former experiments. We clarify the relation between our indirect interaction mechanism and the variant of the public goods game in these experiments. Moreover we interpreted the results of these experiments in the context of our indirect interaction mechanism (Subsection 5.2.4). Our final section (Section 5.3) summarizes and provides the conclusions. We also indicate areas that we consider potential for future work.

5.1 Theoretical Analysis

In order to examine some of the properties of our indirect interaction mechanism with respect to peers' incentives we choose a game theoretical approach. Simply explained our approach is to provide a game theoretical model (a game) for our mechanism which captures both the properties of the standard BitTorrent protocol and that of our mechanism at the same time. We can derive hypothesis from the model and examine other properties of the model such as its relevant solution concepts. Such derivations will allow us to rationalize about the incentives and actions of peers (in the game theoretic sense). However, the validity of the model and the derived rational choices need to be confirmed which we approach by simulating the mechanism (see Chapter 6.

5.1.1 Basics of the Game Theoretical Model

As mentioned before we are looking for a model that, as a first step, can explain why phenomena such as the clustering of similar bandwidth peers, the possibility to freeride, etc. are observed in the standard BitTorrent protocol. In other words the models predictions should at least be in line with what other bodies of work around the BitTorrent protocol have observed and predicted. In order to present this model let us first reiterate some of the basic properties of the BitTorrent protocol from Chapter 2 but from an 'expectation' point of view.

As explained before, BitTorrent peers interact with each other in two ways. Firstly, they can use one of their regular unchoke slots to upload pieces to other peers.[1] When a peer uses a regular unchoke slot it means that it is expecting to receive some contribution in return (tit-for-tat) but it actually has no way of knowing in advance whether this will happen.[2] If the average aggregated upload from the assigned unchoke slot meets the standards of the other peer, reciprocation will occur. Secondly, a peer can use an optimistic unchoke slot to upload a piece to a peer. Every peer has only a single optimistic unchoke slot which it can use. Once it assigns that slot to a peer it is not necessarily expecting to receive a piece in return. That is, the peer is exploring its options in hope of finding a peer that is willing to reciprocate, or in other words a peer who's standards are met with the current transfer rate. Once such a peer reciprocates, the original peer will consider interacting with that peer through a regular unchoke slot as long as the second peer is uploading with a high enough bandwidth to meet the original peers' standards. The standards of each peer are adjusted with respect to the type of interaction that they have with neighboring peers. That is, a peer that is receiving poor service from its neighbors is expected to have a low standard and a peer with good service a high standard. Furthermore, peers are in control of what pieces they receive from other peers with which they are interacting. The general principle in the BitTorrent protocol is that peers prefer to receive rare pieces of the content that are bartering for.

Several key aspects of this short explanation of the BitTorrent protocol have directed us towards our choice of model. The first aspect of the protocol is the uncertainty that peers face regarding their future interactions. That is, a BitTorrent peer never knows whether it has uploaded with a high enough bandwidth to another peer in order to be reciprocated. This means that under standard game theoretical assumptions (rationality and utility maximization) BitTorrent peers would have to be maximizing their *expected* utility. As a result,

[1] Peers use only a single slot to upload to a peer. It is not common to use two or more slots to upload to a single peer.

[2] The peer could try to predict whether it will be reciprocated but the actual choice of the other peer is private information that the original peer is unlikely to be able to acquire.

our model is one that is based on a peer's expected utility. A second aspect of the BitTorrent protocol is the fact that peers interact in rounds and once a round is over they will assess and readjust their strategies. Therefore, our model accordingly characterizes the utility of each player in every round of interaction. The third aspect of the protocol is the fact that a peer prefers to interact with peers that have a higher upload bandwidth and also peers that have rare pieces of the file. We capture these aspects of the BitTorrent protocol in our model which is based on a proposed framework by *Buragohain et al.* [7].

We characterize the upload contributions that a peer makes to its neighbors as a vector in which all neighbors are indexed by some scheme and the values that appear at each index characterize the amount of bandwidth that was contributed to the peer with that index. The notation $\vec{b}_i$ denotes this characterization for a peer i. We have $\vec{b}_i = (b_{i0}, \ldots, b_{i\|N_i\|})$ where N_i is the set of peer i's neighbors. Since BitTorrent peers have a preference for rare pieces we should consider that each contribution that is made to a peer has a certain value for that peer based on the rarity of the piece. That is, each contribution b_{ji} form a peer j to i has a certain value for peer i. This information can be derived from the knowledge that a peer i has about the pieces that its neighbors possess. We characterize this preference with a factor v_{ij} which denotes the value that peer i assigns to a received contribution from peer j. The value of each v_{ij} is known to peer i because it is in control of what pieces it acquires from its neighbors. Notice that we have implicitly assumed here that a peer cannot strategically refuse contribution to another peer once a peer has requested the contribution. That is, if the requesting peer has been a good contributor the serving peer cannot withhold contribution. The only way a peer can strategically refuse contribution is to strategically misreport the contributions that it can make. We characterize the contributions that each peer is able to make to its neighbors as reporting the fraction of the pieces of the file that it possesses. Based on the pieces that a peer i owns and the pieces that a neighboring peer j owns the notation α_{ij} represents the fraction of the file pieces that i can provide to peer j. If α_{ij} is relatively large there is a higher chance that peer i has a rare piece in which j is interested. However, the rarity of a piece is not the only factor influencing the choice of which peer a contribution goes to. Recall that the amount of contribution that a peer has made to a neighbor in the previous round also determines the choice. We have defined a probability function that corresponds to this fact. We denote by $P(\alpha_{ij}, b_{ij})$ the probability of a peer i receiving a contribution from a peer to which it has made a bandwidth contribution of b_{ij} while it has a α_{ij} fraction of the pieces to provide to j. We will refer to this probability as the *probability of reciprocation*. As a result of what we have just explained we can assume that the probability of a peer receiving a contribution from a neighbor is a monotonically increasing function in both α_{ij} and b_{ij} and greater than zero (except for when a peer owns the entire file). In short, more contribution results in a higher chance of reciprocation; and since the more pieces a peer posses the more likely it is to have a rare piece that one of its neighbors is interested in, more pieces also results in a higher chance of reciprocation. Notice that this means that peers are not necessarily better off by increasing their upload bandwidth or increasing their contribution to a neighboring peer because the function is monotonically increasing (Non-decreasing) (see [30, 42] for examples). Table 5.1 summarizes the parameters of our model.

The strategy of a peer i in our game theoretical model is equivalent to the vector $\vec{b}_i$

Table 5.1: Parameters of the Model.

N_i	The set of peer i's (n)eighbors.
$\vec{b}_i$	The vector describing the (b)andwidth contributions that peer i has made to its neighbors (i.e. $(b_{i0},\ldots,b_{i\|N_i\|})$). b_{ij} denotes the bandwidth that peer i assigned to peer j. We denote the aggregate bandwidth that peer i used to upload by $\|\vec{b}_i\|$.
v_{ij}	The (v)alue of a contribution from peer j to peer i that peer i assigns for herself.
α_{ij}	The fraction of the total pieces that peer i can provide to peer j.
$P(\alpha_{ij},b_{ij})$	The (p)robability of peer i receiving a contribution from a peer which it has made a bandwidth contribution of b_{ij} to while it has a α_{ij} fraction of the pieces to provide to j.

where i has to choose how much of its bandwidth to contribute to the other peers (b_{ij}). Naturally the sum of peer i's bandwidth contributions to its neighbors cannot exceed its upload capacity. We will also use the common notation $\vec{b}_{-i}$ to denote the strategy of peers other than i. We assume that peers do not strategically misreport their possessions to influence the value of α_{ij}. Nevertheless, such strategies could easily be included into the model. Given the parameters from Table 5.1 and our notation we describe the expected utility of a peer i in our model of the BitTorrent protocol as follows:

$$u_i(\vec{b}_i,\vec{b}_{-i}) = -\|\vec{b}_i\| + \sum_{j\in N_i}\left[P(\alpha_{ij},b_{ij})\times v_{ij}\times\|\vec{b}_j\|\right] \tag{5.1}$$

Equation 5.1 states that at each round of the game a peer i will decide on how much contribution to make to its neighbors and will expect to loose the same amount of utility for this choice ($-\|\vec{b}_i\|$). On the other hand, it expects to receive some contributions back from its neighbors (the second term in the equation) which is dependent on the probability parameter $P(\alpha_{ij},b_{ij})$, the value that i assigns to receiving the contribution (v_{ij}) and the bandwidth at which it receives the contribution. Notice, that i's utility is dependent on the aggregate contribution of each of its neighbors and that it only needs to know about the total sum of each of its neighbors contributions ($\|\vec{b}_j\|$) in contrast to knowing who the contributions where made to. That is the peer i only needs to know the bandwidth of its neighbors. This is in line with the way BitTorrent works because i already has some idea about the bandwidth of its neighbors that it has already interacted with in the past. We assume that i has some belief about the bandwidth of the peers that it has not already interacted with (i.e the average upload bandwidth in the swarm) which is often available in many BitTorrent clients. [3] In short, peer *i* has a general idea about the upload capacity of its neighbors and it expects to receive some portion of this bandwidth ($P(\alpha_{ij}, b_{ij}) \times \|\vec{b}_j\|$) through direct TFT interaction or through optimistic unchokes.

According to Equation 5.1 a peer has to either increase its contribution or the fraction of the pieces that it can provide to its neighbors in order to increase its utility. This is a well known fact about the BitTorrent protocol.

Notice that up to this point we have presented our model for the standard BitTorrent protocol and demonstrated how the model is capable of explaining the basics of the protocol. In order to further validate our model we also consider explaining some well known phenomena such as freeriding and the large view exploit with the model to account for such observations of the protocol properties.

Explaining Well Known Phenomena

One of the most well known phenomena of P2P systems (and BitTorrent) is the occurrence of freeriding. Accordingly, our model suggests that a peer need not contribute any bandwidth to its neighbors to be able to derive utility. This is because the probability of receiving a contribution from a neighbor is greater than zero as mentioned before. It is due to the fact that peers optimistically unchoke each other. Therefore, a peer could wait to receive optimistic unchokes and download an entire file without contributing any bandwidth (freeriding). In fact the BitThief client is based on the exact same principle [31].

A second well known phenomenon of the BitTorrent protocol is the large view exploit [50] which our model is able to predict. According to our utility function there is a second way that a peer can increase its utility. The second way is for the peer to increase the number of neighbors that it has, which is known as the large view exploit. In practice a large view exploit is possible because a non standard BitTorrent client is able to periodically request new peers from the tracker without any restriction and thereby increasing its chances of optimistic unchokes.

A third and more recently demonstrated phenomenon of the BitTorrent protocol is the idea of strategic piece revelation [30]. Since peers are interested in downloading rare pieces first, any peer could attempt to strategically reveal the pieces that they currently posses in order to prolong the interest of their neighbors to interact with them. While peers could also lie about their possessions, it is more likely that their neighbors will stop interacting with them once they request a piece that has been untruthfully revealed. Therefore the first approach to strategic piece revelation is more likely to succeed. *Levin et al.* have implemented an demonstrated this idea [30]. Again our model is able to explain this phenomena because the probability of reciprocation is a monotonically increasing function. That is, in areas of the reciprocation probability function where an increase in α_{ij} does not increase in the probability of reciprocation and therefore a peer need not reveal the extra pieces.

A forth and final phenomenon that we consider is the principle behind the BitTyrant client [42] which Piatek et al. and Levin et al. [30] both explain. In short, the unchoking slots of a peer could be viewed as an auction for several items in which to win one item, a

[3] The BitTyrant client [42] actually uses a similar assumption which enables it to outperform standard BitTorrent clients.

peer would only require to place a bid that comes last in the winning bids. The BitTyrant client attempts to converge its bids to this value. Our model is again capable of explaining this phenomena because the probability of reciprocation is a monotonically increasing function. That is in areas of the function where an increase of b_{ij} to $b_{ij}+\varepsilon$ does not increase the probability of reciprocation a peer can place a bid for a slot that comes in last by choosing to upload with a bandwidth of b_{ij}.

Having presented our model and described how it is capable of modeling the BitTorrent protocol along side some well known phenomena of the protocol, we next move on to extending this model to our indirect interaction mechanism.

5.1.2 The Indirect Interaction Mechanism Model

We now move on to explaining how our game theoretical model of BitTorrent can be extended to include our proposed indirect interaction mechanism in which peers follow a TFT strategy to generate data flow in cycles of interacting peers. For simplicity we will only present the model of our proposed mechanism for cases where peers are limited to cycles of length 2, but our model and analysis are generalizable to cycles of greater length. So let us consider a peer i that is multi-swarming in two swarms in which the files α and α' are being exchanged. Let us assume that i has received a set of cycles C from the tracker and it decides to upload content to these cycles. That is, i knows about $\|C\|$ peers with which it can form cyclic interactions of length 2. Since i has requested cycles from the tracker it will have received less normal peers in comparison with the situation in which it has not. Nevertheless, the total number of peers with which i can interact is the same as if it had not requested cycles. We denote the set of peers that i can interact with in cycles also by C and the set of normal peers by $\hat{N}_i$, where $\|\hat{N}_i\|+\|C\|=\|N_i\|$. Notice that this means that $\hat{N}_i < N_i$ which, results in a lower expected utility from the second term in Equation 5.1 . We denote the contribution that i makes to each cycle by b_{ic} for $c \in C$. Since i has a limited number of upload slots this means that the amount of contribution that i makes to its neighbors in N_i (in comparison) is decreased. As just explained, this in turn lowers the utility that i acquires from the second term in the utility function. However, the contribution that i makes to the cycles creates a third term in the utility function as follows

$$u_i(\vec{b}_i,\vec{b}_{-i}) = -\|\vec{b}_i\| + \sum_{j\in\hat{N}_i}\Big[P(\alpha_{ij},b_{ij})\times v_{ij}\times\|\vec{b}_j\|\Big] + \sum_{c\in C}\Big[P(\alpha'_{ic},b_{ic})\times v_{ic}\times\|\vec{b}_c\|\Big] \quad (5.2)$$

Notice that peer i is making the same amount of contribution as before. That is, in total a contribution of $\|\vec{b}_i\|$. The difference is that i has only switched the contributions that it previously made to some of its neighbors to peers with which it has formed cycles of length 2. This means that in the eyes of the other peers nothing has changed about the strategy of peer i because they derive expected utility based on the bandwidth of peer i and have no knowledge about which peers i's contribution have gone to. In other words, i has chosen as its strategy to contribute in cycles. The interpretation of the third term in the equation is exactly the same as before. As before the expected utility that i derives from interacting in cycles is dependent on the value that it assigns to pieces of the file that it acquires through this interaction (v_{ic}), the bandwidth of the peers that it is interacting with in cycles ($\|b_c\|$) and the probability of reciprocation ($P(\alpha'_{ic}, b_{ic})$). Notice that this time however, the probability of reciprocation depends on the fraction of pieces of the second file (α'_{ic}) that peer i can provide to the peers in C. That is, peer i is providing pieces of the file α' in return for pieces of the file α. In short, Equation 5.2, which also incorporates the expected utility derived from indirect interaction in cycles, constitutes our model for the proposed mechanism.

The important question is: *When will peer i benefit from switching its contributions from some of its neighbors to cycles or whether it has benefited at all if it does so*. This question leads to the analysis of the model which follows in the next subsection.

5.1.3 Analysis of the Model

For the sake of simplicity we analyze our model in a homogeneous environment to understand some basic properties of the proposed mechanism. By a homogeneous environment we mean one in which all peers have the same bandwidth capacity. The analysis of the model for a heterogeneous environment and including freeriders is likely to be much easier through simulations which we present in Chapter 6. We also assume that there are no freeriders among the peers.

In order to explain some of the properties of the mechanism we consider a peer i at some stage of its download which has a strategical choice between contributing bandwidth in cycles (from a point on) and the choice of contributing bandwidth to only normal neighbors (*i.e.* using the indirect interaction mechanism *vs.* using not using the mechanism). As before, we have $\|\hat{N}_i\| + \|C\| = \|N_i\|$ in the two situations that i might find herself in. Notice that we are assuming that peer i is able to form cycles and that it is able to exchange pieces with the peers in C. If we compare the expected utility of i in these two situations we have

$$u_i(\vec{b}'_i, \vec{b}_{-i}) - u_i(\vec{b}_i, \vec{b}_{-i}) =$$

$$\sum_{c \in C} \left[P(\alpha'_{ic}, b'_{ic}) \times v_{ic} \times \|\vec{b}_c\| \right] - \sum_{j \in N_i - \hat{N}_i} \left[P(\alpha_{ij}, b_{ij}) \times v_{ij} \times \|\vec{b}_j\| \right] \tag{5.3}$$

where $\vec{b}'_i$ depicts a peer's option for indirect interaction in cycles and $\vec{b}_i$ the option for following standard BitTorrent interaction. Obviously i would have gained additional utility from contributing to cycles if the value of Equation 5.3 is positive and it would be indifferent to the choice if the value of the equation is equal to zero.

Since we have assumed all peers to have the same bandwidth capacity, it turns out that if i has a larger fraction of the file to upload to peers in C than it has to upload to peers in $N_i - \hat{N}_i$ (*i.e.* $\alpha'_{ic} > \alpha_{ij}$), and i has a greater value for pieces that it can acquire from peers in C (*i.e.* peers in C have more rare pieces than peers in $N_i - \hat{N}_i$), Equation 5.3 will have a positive value. The reason is that the probability of reciprocation is a monotonically increasing function. Therefore, if $\alpha'_{ic} > \alpha_{ij}$, we have $P(\alpha'_{ic}, b'_{ic}) \geq P(\alpha_{ij}, b_{ij})$. [4] On the other

[4] For any neighboring peer that i decides to unchoke regardless of whether it is normal neighbor or a neighbor that formed a cycle, we have $b'_{ic} = b_{ij}$ because all unchoke slots of a peer have the same bandwidth and i splits its bandwidth equally among the peers that it contributed to.

hand, Equation 5.3 would equal zero if i has the same fraction of pieces to give to peers in $N_i - \hat{N}_i$ as it does to peers in C and i assigns the same value to pieces that both sets of peers provide (i.e. $v_{ic} = v_{ij} \ \forall c, j \in C, N_i - \hat{N}_i$).

So imagine a peer i which is leeching the file α and seeding the file α' (i has a leecher-seeder status) and on the other hand a peer j which is doing the exact opposite (j has a seeder-leecher status). According to Equation 5.3 these two peers would gain additional utility by interacting indirectly in cycles. That is when such pairs are coupled together the fraction of the file that the pairs can provide to each other are maximal. Because this fraction is at its maximum it would follow that the value that each peer assigns to the contributions from peers in cycles is greater than the value that it assigns to contributions from normal peers ($v_{ic} > v_{ij}$). Hence, Equation 5.3 would have a positive value in this situation. Notice that if peer i already knows that peer j has chosen the option to interact indirectly with peer i, peer i would also decide to interact indirectly with peer j because it would derive a greater utility than choosing to follow the standard BitTorrent interaction. The same argument is valid in reverse for peer j due to the symmetry of the situation. As a result, under our current assumptions it is a *Bayesian Nash Equilibrium* to choose for contributing to cycles.[5] Even if i (or j) has the same value for pieces that it can acquire through cycles and pieces that it can get through normal interaction, the two terms of the equation cancel out in worst case. That is i and j would at worst be indifferent to choosing either option. This argument holds for any number of leecher-seeder/seeder-leecher pairs and leads to our first hypothesis regarding the properties of the mechanism:

Hypothesis 1. ***(Leecher-Seeder/Seeder-Leecher pairs are better off interacting indirectly)*** *A multi-swarming peer i that is seeding file α' and leeching file α and a multi-swarming peer j that is doing the exact opposite will be better off to interact in cycles in comparison with when they both follow the standard BitTorrent protocol or when they do not seed the file which they possess entirely.*

A second hypothesis that can be derived from Equation 5.3 is regarding the bootstrapping period of a peer. Again consider a peer i that is partially or wholly in possession of a file α and seeking to acquire file α' and another peer j with the opposite status. At this stage of a download, $\alpha'_{ik} = 0 \ \forall k \in N_i$ (α is at its minimum) which means that the utility that peer i expects to derive from the second term of Equation 5.2 is at its lowest. At this stage, i would have to wait to receive contributions as optimistic unchokes from its neighbors in N_i (or $\hat{N}_i$) before it can start bartering pieces of the file α'. On the other hand peer i possesses pieces of the file α that it can barter for pieces of the file α' with peer j. This means that i would no longer have to wait for the optimistic unchokes if it interacts indirectly with peer j. By symmetry the same argument holds for j. As a result interacting indirectly would achieve a higher utility for both peers at bootstrapping phase because there is a higher chance that contributing in cycles will acquire reciprocation for both involved parties ($\alpha_{ij} > 0$ for peer i and $\alpha'_{ji} > 0$ for j). Hence we have the following hypothesis:

[5] This situation constitutes a Bayesian Nash equilibrium because peers hold beliefs about the bandwidths of their counter parts.

Hypothesis 2. ***(Faster Bootstrapping)*** *A multi-swarming peer will be able to bootstrap faster if it interacts indirectly in cycles in comparison to when it follows the standard BitTorrent bootstrapping procedure.*

Notice that up to this point we have only covered very specific combinations of multi-swarming peers interacting indirectly. Nevertheless, other combinations of the peers and values for the parameters are also possible such that they lead to a non-negative value for Equation 5.3. However, such combinations and values are less structured than the leecher-seeder/seeder-leecher pairs or the bootstrapping case. These combinations are more difficult to derive analytically from the model. That is, leecher-leecher pairs in cycles (or any other combination) can also lead to a positive difference in utility but it is more difficult to derive a single coherent hypothesis of what will happen as we can for the case of leecher-seeder/seeder-leecher pairs for example. In such cases, the probability of reciprocation and the assigned value for each contribution from each peer have a more prominent role in determining the value of Equation 5.3. However, note that by expectation interacting indirectly with a multi-swarming peer is similar to interacting with any other random peer in the set of neighbors. That is, we expect the fraction of pieces that the two multi-swarmers are able to barter and the values that they assign to each others pieces be similar to those of a random peer in the set of their neighbors. Hence we can derive the following hypothesis regarding the other possible combinations of indirect interaction:

Hypothesis 3. ***(The indirect interaction mechanism will not reduce the performance of the standard BitTorrent protocol)*** *The standard BitTorrent protocol will not significantly outperform the indirect interaction mechanism in terms of the time required to download an entire file if other combinations of multi-swarmers choose to interact indirectly rather than follow standard protocol. By 'other combinations' we mean scenarios of indirect interaction that have not been covered in the previously mentioned hypotheses.*

In conclusion we expect to see a higher utility for a peer at bootstrapping phase in comparison with the standard bootstrapping procedure. In addition we also expect to see a higher utility for a peer that can form *enough* leecher-seeder/seeder-leecher pairs during its download process; where the term 'enough' can be interpreted as having the ability to exchange pieces indirectly for the duration of a download or the ability to acquire more rare pieces during the download from cycles . Moreover, in the long term we expect to see only small differences in utility if non of the previously discussed scenarios occur (*i.e.* when assuming a heterogeneous population, other combinations of pairs, *etc.*). We will partially validate our hypotheses in our next chapter where we describe our simulation results of the mechanism in various settings. However, before we move on to presenting the results of the simulations, we find it necessary to also present a second model of our mechanism which aims to hypothesis about how humans would react in the face of our indirect interaction mechanism. This second model takes an *experimental game theory* approach to studying the properties of the mechanism.

5.2 Experimental/Evolutionary Analysis

One important aspect of the indirect interaction mechanism is that it is intended to be used by human beings which are in control of their BitTorrent clients. As you may recall in our analysis of our game theoretic model for the mechanism we have made certain assumptions which are regarded as classic game theoretical assumptions. Such assumptions include rationality and utility maximization in the game theoretic sense. That is, the players that are involved in a game (in our case people that are represented as BitTorrent peer(s)) are regarded to be fully rational, meaning that they have the ability to make the *best* possible choice of strategy when faced with multiple choices of possible actions. Further more the criteria by which the desirability of an outcome is judged (i.e. the best choice) is determined by the choice(s) that results in the most amount of utility for the player. As mentioned, such assumptions are classic assumptions of game theory and depending on the environment or the game, certain additional assumptions regarding matters such as the form of the utility function of each player, whether players are risk taking or risk averse and the amount and type of information that each player has, may accompany the first two assumptions. In what follows we present a second model of the indirect interaction mechanism which is based on an experimental game theory approach that we argue is useful and perhaps necessary. Such approaches addresses some of the critiques regarding the classic game theoretical assumptions the main one of which questions the validity or rather practicality of these assumptions when human beings are involved in game play.

5.2.1 A Second Analysis, Why?

Perhaps the most important critique to classical game theory is the argument of *bounded rationality* [49] which certainly holds when human beings are involved in making strategic choices. In short, bounded rationality states the fact that the ability of human beings to make choices is limited by the amount of information they have, their cognitive abilities and the limited amount of time they have to make their decisions. Perhaps the most important consequence of this fact is that it underpins the validity of the classical rationality assumption in game theory when human beings are involved in game play. While this is certainly the case in our environment of study it is important to remember that it does not invalidate the results that we or others have obtained under these assumptions. Furthermore, there exist arguments in favor of the classical assumptions such as the argument that on average or in the long run the choices of a population of players would converge towards the choice(s) that a population of fully rational players would make. Moreover, there are many games in which the bounded rationality of the players does not affect the outcome that is predicted under classical assumptions. As *Simon* states, perhaps a useful way of thinking about this matter is to think of the first approach as how a player '*should*' act while to think of what we present in this section as how the players '*would*' act [49]. It is logical to assume that the closer the assumptions are with the underlying properties of the environment and players, the more resembling the 'should's and 'would's become. At this point however, we do not feel that this is the right place to argue about the nature of the different approaches to modeling behavior. Instead, we would like to familiarize the reader with some of the complementing approaches and also present some arguments to encourage future research in this field to consider a combination of the possible approaches in order to be able to provide better predictions of what should and what would happen when human beings are involved.

In order to address and consider the argument of bounded rationality at least two complementary modeling approaches have been taken. The first approach which we shall refer to as the *experimental* approach is used in the field of experimental psychology and involves the design of carefully crafted lab experiments with actual human subjects. Usually in the experimental approach, subjects are required to play a game (the game that is considered to be a model of the environment under study) the rules of which are presented to the subject before playing the game. Depending on the environment and properties that are to be studied, certain limitations are placed on what the subjects are allowed and not allowed to do. For example, subjects might (or might not) be allowed to communicate before or during the game, subjects might (or might not) be given information about other subjects and they might (or might not) be allowed to repeat the game with the same or a different group of subjects, etcetera. One of the advantages of an experimental approach is that it could closely match the underlying properties of the environment and the players. One of the disadvantages is that such experiments take a lot of time and effort to prepare and run and it is very difficult to conduct such experiments with large groups of subjects and at the same time control the variables of the experiments.

On the other hand, a second approach which we shall refer to as an *evolutionary* approach that is more popular in the field of evolutionary biology. This approach also addresses the bounded rationality issue by assuming that players are not perfectly rational but rather assumes that players choices (could) converge towards a more rational choice of action in a process similar to an evolutionary process. In the evolutionary approach the game (model) is usually simulated with a certain number of players and a certain possible set of strategies for the players. The strategies are distributed between players according to some a priori distribution. The evolutionary process may or may not allow players to change their strategy (type) through either a conscious choice or mutation. For example, players might generate offspring in the evolutionary process that have an identical type to them or mutate to have a different type or even change their own type by copying the type of other players that have been more successful. One of the advantages of evolutionary approaches is that they can be useful for understanding the properties of strategic environments with a large population and in a repeated setting. On the other hand the way the evolutionary process is simulated can have large effects on the observed outcome (*i.e.* mutation rate, initial distribution of strategies, *etc.*). Nevertheless both experimental and evolutionary approaches address the bounded rationality argument to some extent by complementing the classical approach with a less cognitively demanding set of assumptions for the players.

One of the popular games that has been widely worked on with the aforementioned approaches is the Public Goods Game (in short PG) which can be considered as an extended Prisoner's Dilemma (PD) game (see Chapter 2). In our following subsections we will explain how we can approach this game as a model of the BitTorrent protocol and to provide an experimental view on the indirect interaction mechanism with the PG game.

5.2.2 The Public Goods Game and its Relation with BitTorrent

The public goods (PG) game in its standard form is a multi-player game that captures the social dilemma between selfish interests and the interests of a group and is considered as an extended prisoner's dilemma game [14]. As a result the PG game can be regarded as a general model for P2P systems and BitTorrent in particular because it captures the dilemmas that exist in P2P systems.[6] In its standard form (symmetric linear PG game) the game consists of n players which have been bestowed some amount of private good (also referred to as tokens) T_i; $\forall i \in \{1,\dots,n\}$ which are usually in the form of currency in experiments. Players can keep or invest their private good in some public good (or public project). The public good is represented by a public pot in which tokens are deposited. The amount of contribution to the public pot can range from zero to all available tokens. All players have to decide simultaneously on the amount of contribution that they would want to make to the public pot. In other words, the strategy s_i of a player i is drawn from the range $[0, T_i]$. Once the players have made their contributions, the amount of tokens in the public pot ($\sum s_i$) is multiplied by some factor $1 < \gamma < n$ and divided equally among all players as a payoff for their investment in the public pot. The utility of each player could be described as follows:

$$u_i(s_i, s_{-i}) = -s_i + \frac{\gamma}{n} \sum_{i=1}^{n} s_i$$

As before s_{-i} denotes the strategies of other players than i. Note that the utility of a player is linear and that changing the names (indexes) of the players does not have any effect on the form of the utility function, hence the name linear symmetric PG game. Both one-shot and repeated variants of the game have been described in the literature as the standard form.

Despite the popularity of the standard linear form of the PG game many other forms of the game exist. For example one may require that the strategies of the players to be drawn from a range of numbers that are quantifiable or that they be limited to some other bounds. In another form which is referred to as the threshold PG game no public good is produced unless the amount of contributions exceeds some threshold α. Despite all forms of the game, *Ledyard* approaches the game from a more general point of view and states that it can be described as a game in which the amount of public good that is generated is determined by some function $y = f(t)$ where t is the amount of tokens used to produce y amount of public good. The environment of the game is then described as $g =< f, u_1, \dots, u_n, T_1, \dots, T_n >$ [27]. He states that virtually any public good or social dilemma experimental environment is a special case of g in which specific forms for $(f, u_1, \dots, u_n)$ and specific values for $T_1, \dots, T_n$ are chosen [27]. The utility of a player i in the general form can be described as a function of i's strategy and the aggregate contributions of players to the public pot: $u_i(s_i, f(\sum_{i=1}^{n} s_i))$. Note that the initial amount of tokens and the utilities of the players are not necessarily the same. *Ledyard* also points out that it is generally accepted that all forms of the game have similar properties or that rather similar changes in different games have similar effects on the outcome. He also states that this statement is rarely tested [27]. Throughout this study we presume this statement to hold. As a result, BitTorrent can be modeled as a general form

[6] We will elaborate further on the reasons later in this subsection.

public goods game in which the public good that is produced is a function of the amount of private good that is contributed where peers have different amounts of initial tokens and that they derive different utilities from participating in the system. Perhaps it is useful to think of tokens (private good) as files (or fractions of files) and the bandwidth that peers possess in the context of the BitTorrent protocol. This means that BitTorrent peers initially possess some amount of pieces which they can provide at certain bandwidth. This corresponds to the players initial tokens which they can invest. On the other hand the total amount of file pieces that are invested (with various bandwidths) into the public pot are then distributed according to some function between the peers and they derive utility.

There are generally two types of predictions regarding the PG game. The first prediction derived from classical game theory predicts that players will not contribute anything to the public pot because no matter what the other player's actions are, for each token contributed the contributor will yield a return of $\frac{\gamma}{n} < 1$ [27, 15]. On the other hand if all players contribute all of their tokens they would be better off. This property of the game constitutes the social dilemma in the PG game. Classical game theory predicts that it is a dominant strategy to contribute 0 tokens and that players would '*freeride*' on the efforts of others because it is the rational choice. A second prediction which is based on social psychology predicts that all players will contribute something due to properties such as altruism, social norms, group identification, etc. [27]. However, both models turn out to be incorrect and not generalizable to humans as many experiments show that some players will freeride, some will contribute a portion of their tokens and others will contribute all of their tokens.

The wide range of possible outcomes observed in PG games suggest that there are other variables that have an effect on the outcome of the game and hence many controlled experiments have been conducted to identify and investigate such variables. Table 5.2 taken from [27] summarizes some of the variables that have been studied. The general observed outcome of the PG game is that players do contribute some tokens but that contributions decline significantly in subsequent rounds of the game.

Let us elaborate further on Table 5.2. *MPCR* represents the marginal return of utility for investing 1 token in the public pot. In the case of the linear symmetric PG game this is equal to $\frac{\gamma}{n}$. According to Table 5.2 increasing γ (or the amount that the total investment is multiplied by) will have a strong positive effect on the percentage of contribution to the public pot. Furthermore, the number of players that are involved in the game seem to have no effect on the contributions. This result is dependent on the results of experiments in which it was found that MPCR matters and n (the number of players) does not. Another interesting variable of the experiments, *repetition*, is shown to have a strong negative effect on the contributions of players. This means that if the game is repeated with the same test subjects for several times, the percentage of contribution drops as the game progresses. The next variable, *common knowledge*, constitutes ex-ante knowledge of the distribution of types of players and according to Table 5.2 has a relatively positive effect on the contribution of players. *Thresholds* effectively result in no generation of public good if contributions do not reach a certain threshold. These also seem to have a positive effect on the contributions. Apart from the aforementioned variables which are easy to control in experiments several other variables including beliefs, experience and friendship have also been investigated which are harder to control. According to Table 5.2, beliefs about other subjects and

Table 5.2: Variables affecting the percentage of total contribution to optimal contribution in the PG game.

Environment	**Easy to Control**
	% Contribution
Marginal Per Capita Return (MPCR)	++
Number of Players	00
Repetition	- -
Common Knowledge	+
Thresholds	+
Systemic	**Difficult to Control**
	% Contribution
Beliefs	+
Experience	-
Friendship/Group Identification	+
Design Variables	
	% Contribution
Communication	++

A +/- sign means positive/negative effect on the % of contribution. A 0 sign means no effect. A repetition of a sign means a strong and repeatable effect while a single sign means that it is hard to replicate the results in experiments.

friendship both have a positive effect on the percentage of contribution while experience in the PG game has a negative effect. Finally relevant communication between subjects in small groups prior to game play (decision regarding the design of the experiment) has been shown to have a strong positive effect on contribution. This is regardless of whether communication is verified or not and it can lead to information regarding the contributions of others or group identification. On the other hand it is not known why this effect exists and what the effects are for large groups. We refer the reader to work of Ledyard [27] for more detailed information.

With respect to the BitTorrent protocol similar effects have been observed. For example *Zghaibeh et al.* have observed an increase in freeriding behavior in BitTorrent which they attribute to repetition and the experience that people have gained with the BitTorrent protocol [53]. Another example is the effect of group identification which can be observed in combination with reputation and punishment in private BitTorrent communities. It is a well known fact that private communities exhibit larger amounts of cooperation. A third well known fact is the positive effect of reputation systems on contributions in BitTorrent. However, some of the previously described variables are difficult to isolate and control in the online environment of BitTorrent. For example pre-play communication, friendship, repetition and even the number of players are difficult or impossible to control. Furthermore, variables such as common knowledge have not been directly investigated in the context of

BitTorrent, but according to the previous results we can speculate that if BitTorrent peers had some common knowledge about the bandwidth of other peers it would affect their contribution accordingly. On the other hand a variable like thresholds is irrelevant in the context of the BitTorrent protocol.

In conclusion the PG game can be considered as a general model of the BitTorrent protocol according to *Ledyard*'s characterization of the game. Additionally the PG game and the BitTorrent protocol exhibit similar properties as described by Table 5.2. Both BitTorrent and the PG game have been relatively well studied with respect to the effects of the previously described variables and show close resemblance. In the following subsections we will discuss a specific experiment involving the PG game which partially models our indirect interaction mechanism and present the results of these experiments to address the bounded rationality argument and hopefully better predict human behavior.

5.2.3 The Public Goods/Indirect Reciprocity Game as a Model

As explained in the previous subsection we can model the BitTorrent protocol with a PG game with specific values chosen for the initial tokens, a function for the generation of public goods which is dependent on the total contribution and specific utility functions for each peer. while our indirect interaction mechanism follows the same standard procedures of the BitTorrent protocol when peers are not multi-swarming, the indirect interaction of multi-swarming peers adds several features to the protocol which cannot be modeled as a single public goods game. However, a specific type of well studied PG game exists that can partially model our indirect interaction. By partial we mean indirect interaction when peers are restricted to interact in cycles of length 2. This type of PG game is a combination of the PG game followed by a second smaller game which is referred to as the '*indirect reciprocity*' (in short IR) game.[7]

The IR game is very simple and it simply involves a player giving away some tokens to an other player(s). The receiving party can then decide to reciprocate by paying back some tokens to a player(s) that is (are) not necessarily the one(s) that gave the tokens in the first place or, keep the money. This simple game has been combined with the PG game in several studies [35, 40] to consider the effect of indirect reciprocity (or rewards) in the PG game. In these studies the IR game is combined with the PG game as follows:

> *n* players first play a round of a PG game. This round is followed by a round of an IR game in which each player is once in the position of a 'donor' and once as the receiver of donations. Once the IR round is over the process is again repeated from the beginning. Furthermore, player acquire knowledge of the actions of other players in the previous PG round of the game in the form of reputation before playing the IR game.

To understand how sandwiching rounds of the IR game between rounds of PG could model our indirect interaction mechanism, recall our argument that BitTorrent can be modeled as a PG game. On the other hand, the IR game directly corresponds to indirect interaction in cycles because during indirect interaction each peer equally assumes both the role of

[7] The game is also referred to as Reward game in [48].

a donor and a recipient. Therefore, we can consider this setting as a model for our indirect interaction mechanism. However, notice that an important variable in the former game is that players are aware of each others actions during the previous PG game and observe the actions of their counter parts in the IR game. In order to account for this part of the game we are relying on the property that peers can directly observe each others action in cycles of length 2 and will not lie about their multi-swarming activity to the tracker since lying does not result in any benefit for the lying peers. Nevertheless, this does not mean that all indirectly interacting peers can monitor the interactions of all other indirectly interaction peers. In other words the PG/IR game partially models the IIM mechanism.

5.2.4 Outcomes of the PG/IR game

In an experiment conducted by *Milinksi et al.* [35] 19 groups of students were asked to play the PG/IR game. Ten groups played one round of the IR game followed by the PG game in an alternating pattern for 16 rounds. At round 17 every second group was told that from then on only PG games would follow until the end of the game. The 9 other groups played 8 rounds of PG followed by 8 rounds of IR and another 4 rounds of the PG game. Again every second group was told at round 17 that from then on only PG games would follow until the end of the game. The results of the experiment are summarized in Figure 5.1.

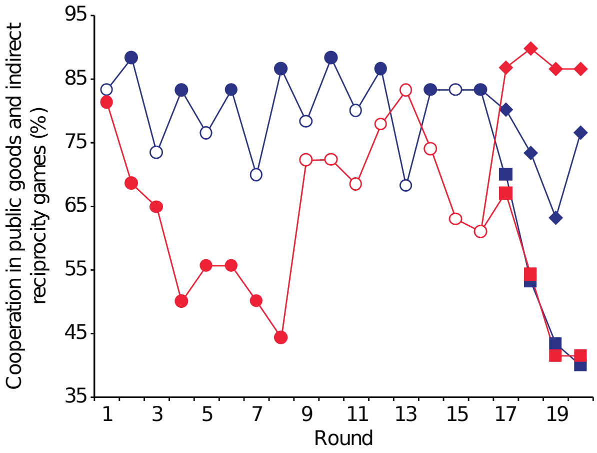


Figure 5.1: Blue Symbols correspond to the first 10 groups and red symbols to the second 9 groups. Filled Symbols correspond to PG games and open symbols to IR games. Square symbols correspond to cases were it was announced that only PG games would follow. Diamond Symbols correspond to cases were there was no knowledge of what type of game would follow [35].

As expected from a regular repeated PG game (the first 8 rounds of red symbols) it can be seen in Figure 5.1 that the contribution of players drop as the result of repetition (or experience). However, surprisingly the same effect does not show in the other groups (blue symbols) that were alternating between PG and IR games. Also note that after the

16th round of the game groups that were not told that only PG games are to follow have maintained a high level of cooperation. *Milinski et al.* attribute this observation to the benefits of the IR game in combination with the information that players have regarding the behavior of others in the previous round [35]. Also note that the results depicted in Figure 5.1 can be interpreted as having a small chance of playing an IR game should maintain the high levels of cooperation observed (at least for 4 rounds.)

This experiment suggests that in the case of our indirect interaction mechanism we should observe similar effects on the contributions of BitTorrent peers. To interpret this result in the context of our mechanism assume that the tokens of the PG game are files or portions of files that peers possess. We make no differentiation between contributing partial files (leeching) and entire files (seeding) because by the time a leecher has acquired an entire file he would have bartered many of its possessed pieces of the file. However we do make a distinction between multi-swarming and leeching/seeding. Multi-swarming peers not only barter a single file but they additionally leech/seed (at least) a second file. Therefore we could consider that they are investing more tokens in the PG game. As a result, multi-swarming peers in effect are worthy of a higher reputation in our indirect interaction mechanism. Also note that only multi-swarming peers are allowed to form cycles (play the IR game) because of the way our indirect interaction mechanism works. This means that only higher reputation players are able to play the IR game and lower reputation peers are automatically filtered out. If these multi-swarming peers are able to interact in cycles (the equivalent of playing an IR game), even if there is a relatively small chance, we should observe a tendency towards indirect interaction according to the results of *Milinski et al.* (if our assumption regarding the reputation of multi-swarming peers holds). This result is in line with our hypotheses from our game theoretical model of the mechanism (See Subsection 5.1.3).

A second study by *Panchanathan et al.* [40] on the PG/IR game takes an evolutionary approach to modeling behavior which reveals additional the properties of the indirect interaction mechanism. *Panchanathan et al.* study the interaction of three basic strategies, *freerider*, *cooperator* and *shunner* in an evolutionary model. According to their model players are not perfectly rational but rather strategies that perform better in terms of the payoff that they generate for their followers (evolutionary fitness) survive and are copied/inherited by offspring that are generated at the beginning of each round of play. A freeriding strategy is one that will never contribute anything while a cooperating strategy is one that will always contribute something in both PG and IR games, even when the receiving party in the IR game has a bad reputation. A shunner is the same as a cooperator with the difference that it will not contribute anything to a party of bad reputation in the IR game. A small chance of error is introduced by which both shunners and cooperators occasionally fail to follow the prescribed strategy (perhaps due to running out of resources). All players start the game with a neutral reputation. *Panchanathan et al.* demonstrate that two evolutionary stable equilibria exist if the reputation requirements that we outlined before exist. Freeriding is always a stable equilibrium. On the other hand if shunners constitute a large enough portion of the population a greater attraction towards shunning strategies exists only if the long term benefits derived from the IR game are greater than the cost for contribution in the

PG game. According to our game theoretical model from Section 5.1 we have reason to believe that this condition is satisfied when leecher-seeder/seeder-leechers pairs interact in cycles of length 2. However, since the benefits of indirect interaction for other combinations of pairs are not clear, we cannot conclude that the indirect interaction mechanism will reach a stable point of cooperation through shunning strategies. Nevertheless it has been pointed out that a strategy that relies on the reputation of a party in only the first round of interaction and on personal experience in all further rounds of interaction (which is essentially how the indirect interaction mechanism works) is a 'good' strategy when social dilemma games (*i.e.* PG) are followed by IR games[38].

As a conclusion for the experiments that we have reported, we expect to see a tendency towards multi-swarming in the context of our indirect interaction mechanism. That is people are expected to show tendency towards using the indirect interaction mechanism in order to be able to use the benefits of indirect reciprocity. However, the long term sustainability of the drive towards indirect interaction is still under question and we cannot draw any conclusions on whether the indirect interaction mechanism will incentivize seeding in the long term.

5.3 Conclusions and Future Work

In this chapter we have analyzed the indirect interaction mechanism that we presented in Chapter 4 from a classical game theoretic view and an experimental/evolutionary view point. In particular we have presented a game theoretic model for the indirect interaction mechanism that demonstrates the benefits of using such a mechanism and how it can address the general seeding incentives problem that exists in BitTorrent. We provided reasons to believe that the indirect interaction of peers in cycles is beneficial from a game theoretic point of view especially when the combination of peers that interact maintain a leecher/seeder status. This benefit was shown by arguing that indirect interactions lead to (weak) Nash equilibria as was outlined in Section 5.1.

Nevertheless a concern was raised that a game theoretical analysis is not able to correctly predict the behavior of humans in strategical situations such as the public goods game. As a result, we considered an experimental/evolutionary analysis that could lead to better predictions of human behavior with respect to our mechanism. We reported several experiments that lead to the conclusion that humans would indeed show a tendency towards indirect reciprocity which is an essential feature of the indirect interaction mechanism. However, the sustainability of this tendency was reported to be dependent and very closely tied to reputation mechanisms and the ability to sustain long term benefits from indirect reciprocity. As we have argued in Section 5.1 situations can occur for which the benefits of indirect reciprocity cannot exceed the costs of participating in the indirect interaction mechanism. Nevertheless, we have also argued that we expect that these costs and benefits to be roughly equal. On the other hand studies suggest that people show tendencies towards conditional cooperation in PG games [21] which is essentially what the indirect interaction mechanism seeks to do. This, in addition to the arguments that we have provided allows us to arrive at the following conjecture:

> ***Conjecture .*** *The indirect interaction mechanism incentivizes seeding (or partial seeding) of files in the BitTorrent protocol.*

Recall that during our analysis we have derived several hypotheses form our game theoretical model. Perhaps further hypothesis can be derived from the model in Subsection 5.1.2, the analysis of which will reveal further properties of the indirect interaction mechanism. Also recall that the analysis that we have provided is limited to certain assumptions such as the homogeneity of the peers and the absence of freeriders. Further work is required to demonstrate how the indirect interaction mechanism will work when such assumptions are not met.

Finally, our attempt to address the bounded rationality argument has been partially inconclusive and we consider future experimental analysis of the indirect interaction mechanism necessary to be able to correctly predict human behavior before fully implementing such a mechanism for the BitTorrent protocol. While experiments suggest that humans are better incentivized by combining punishment mechanisms with social dilemma experiments such as the PG game [48], the fact is that implementing such mechanisms requires demanding reputations mechanisms that greatly add to the complexity of online mechanisms. Furthermore, it is unclear how effective punishments can be induced on asocial behavior in online environments which are highly dynamic and such behavior can occur anonymously. It is likely that rewarding mechanisms such as our indirect interaction mechanism are less complex to implement and that they could overcome the complexity of online reputation mechanisms.

Chapter 6

Simulations

In this chapter we present the results of our simulations of the Indirect Interaction Mechanism (IIM). We introduced this mechanism in Chapter 4 and modeled the mechanism in Chapter 5. We use the results of our simulations to asses the validity of the analytical model and the hypotheses that we derived from the model. We only focus on the first two hypotheses in Section 5.1.3 due to the limitations of our current simulation tool. These hypotheses state that indirect interaction will benefit the bootstrap and download time of multi-swarming peers.

This chapter is organized into six main sections. In our first section we provide the details of our simulation setting (Section 6.1). Our second section is dedicated to comparing our simulations of the Indirect Interaction Mechanism with standard BitTorrent. In this section we focus specifically on the bootstrapping of peers (Section 6.2). The third section is dedicated to comparing IIM and standard BitTorrent with respect to the download time of peers (Section 6.3). In our fourth section we present our results of the efficiency of IIM (Section 6.4). Subsequently we present limited results of our simulations in a heterogeneous setting (Section 6.5). Finally we conclude this chapter in Section 6.6.

6.1 Simulation Setting

In order to verify the hypotheses that we have made in Chapter 5 about the Indirect Interaction Mechanism (Section 5.1.3) we have simulated the Indirect Interaction Mechanism with a modified version of *TriblerSim.* [1] We have modified TriblerSim to simulate a simple setting of Indirect Interaction which involves 2 swarms and 2 multi-swarming peers (one seeder-leecher and one leecher-seeder) which can have a cyclic interaction of the type described in Chapter 4. Figure 6.1 demonstrates the types of scenarios that our tool is capable of simulating.

The tool is capable of controlling the number of leechers/seeders in each of the swarms in addition to their upload bandwidths [2]. We can also control the amount of time that each

[1] TriblerSim is a simulator developed by the Michel Meulpolder at the Tribler[1] group to simulate BitTorrent and BarterCast[34]. It is available at the Tribler group SVN repository.

[2] The upload bandwidth is assumed to be the bottleneck therefore all peers are assumed to have an unlimited

peer is willing to seed a file once it has downloaded an entire file. Communication between peers is assumed to be instantaneous (i.e. no network delay) and the tool lacks a way of modeling the properties of the underlying network. [3] A typical simulation starts with leechers having no pieces of the file unless a *random scenario* option is provided which will randomly assign some pieces of the file to each of the leechers. Finally we can control the size of the files associated with each of the swarms in the simulation. Simulations start with all the peers announcing their presence to the tracker in a random order at time 0 while immediately requesting a list of peers to interact with from the tracker.

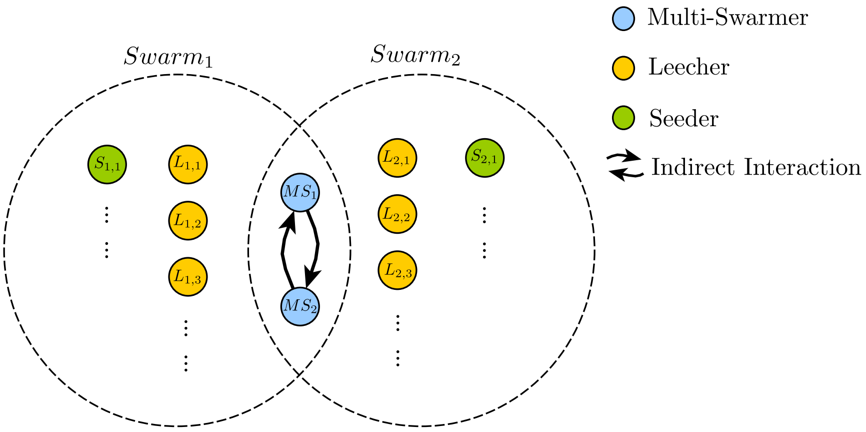


Figure 6.1: Possible simulation scenarios of the Indirect Interaction Mechanism.

Demonstrated results throughout this chapter have been obtained with two identical swarms of 1 seeder and 25 leechers (i.e. a relatively small sized swarm). Seeders have been configured to seed a file for ever while leechers have been configured to disconnect from the swarm as soon as they have finished downloading the file. Additionally there are 2 multi-swarmers that that are present in both swarms (making the total number of peers in each swarm 28). Each multi-swarming peer is leeching one file and is in possession of an entire second file such that the two peers are performing opposite actions and thereby able to form an indirect interaction cycle of length 2. Both multi-swarming peers withhold the contents of their second file from other leechers (i.e. do not seed and are invisible to regular leechers) and only barter its pieces for pieces of the file they are leeching. We will refer to this as *conditional seeding*. Multi-swarmers are also configured to disconnect from the swarm as soon as they have finished downloading the file. In both swarms a 128*MB* file is bartered and all peers have an upload speed of 512*Kbps* (homogeneous environment). [4] Demonstrated results have been obtained with 50 simulation runs in which multi-swarmers

download bandwidth.

[3] Recall that our model of IIM from Chapter 5 also does not consider properties such as as network delays for simplicity.

[4] All other peer characteristics such as the number of unchoke slots and the number of peers that they receive from the tracker are set according to the BitTorrent specifications.

indirectly interact (i.e. IIM is enabled). These simulations are mirrored with the exact same number of runs in which the multi-swarmers do not interact indirectly and act as normal leechers (i.e. two standard BitTorrent swarms each with 27 peers). Unless explicitly stated otherwise, this constitutes the setting in which the results have been obtained.

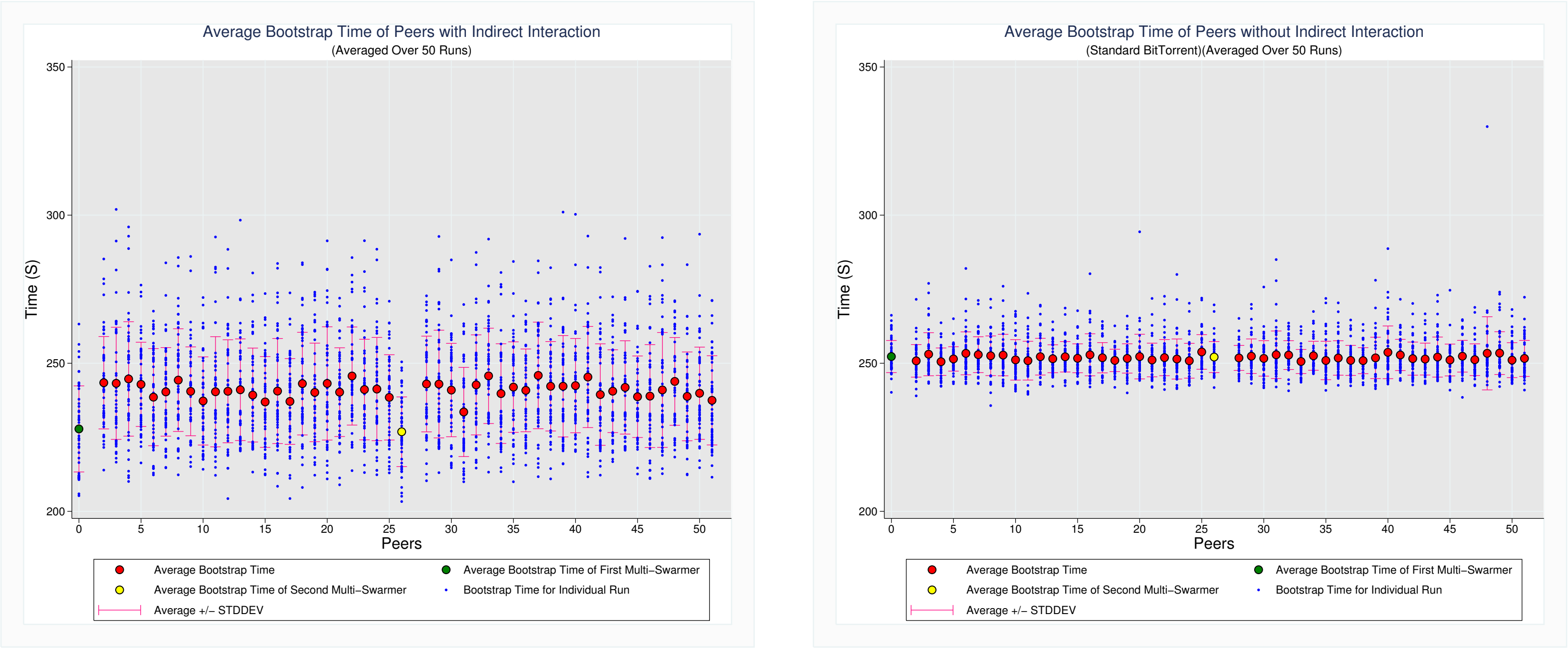
Having explained the setting of the simulations we now proceed with presenting our results by initially focusing on the bootstrapping of peers and comparing the results of the mirrored simulations.

6.2 Bootstrapping with Indirect Interaction

Our first set of simulations is directed towards attempting to reject our hypothesis regarding the bootstrapping of peers. Our notion of bootstrapping in this Chapter is somewhat different than the regular definition of *obtaining a single piece of the file*. Here, we consider a peer to be bootstrapped once it has obtained at least 10% of the file. The reason behind this choice is that it usually takes some time for a standard BitTorrent peer to effectively utilize all of its upload/download slots. Since the exact time at which the peer can fully utilize resources varies, we have chosen a large enough download percentage to make sure that the standard BitTorrent peer has reach this particular point. Therefore we define a peer to be bootstrapped when it has obtained 10% of the file. We further elaborate on this definition later in this section.

Recall our second hypothesis (***H2***) from Section 5.1.3 in which we hypothesized that in a homogeneous environment indirectly interacting peers will bootstrap faster in comparison to standard BitTorrent peers. Based on our model of IIM we speculate that indirect interaction will give multi-swarmers a higher chance of reciprocation because they will have a higher bargaining power when they are not in possession of interesting pieces of a file to offer to others. This means that once a peer has no option to barter in one swarm it can switch to bartering in a second swarm, thereby reducing idle time. Notice that by definition a bootstrapping peer is in such a situation.

In our first set of simulations we are looking for results in which the standard BitTorrent protocol outperforms IIM on average and thereby reject our hypothesis. Notice that a negative result will not only reject ***H2*** but also indicate that our analytical model of IIM is not capable of correctly predicting behavior. Figure 6.2 demonstrates the results of simulating IIM mirrored with simulating standard BitTorrent over 50 runs. The two multi-swarming peers of our setting are indicated by the colors green and yellow. Furthermore, the first 27 peers are bartering in the first swarm while the second 27 peers are bartering in the second swarm.



(a) Bootstrap with IIM

(b) Standard Bootstrap

Figure 6.2: Comparison of the average bootstrapping time of peers in IIM and standard BitTorrent; All peers have an upload bandwidth of 512 *kbps*; A peer is considered bootstrapped if it possesses 10% of the file.

Our results from Figure 6.2 demonstrate that the two multi-swarming peers were able to download 10% of the file faster than the regular leechers in their swarm. Also note the slight decrease in the overall bootstrap time of regular leechers. This can be attributed to some resources of the seeder and other leechers being being freed to regular leechers as a result of the multi-swarming peers engaging in indirect interaction. This effect can be seen more clearly in the next section. Furthermore, notice the low variance of bootstrap times for standard BitTorrent which is what we expect to see in our simulation. [5] Even though the multi-swarming peers have bootstrapped faster than their counter parts what we are testing to see whether the two multi-swarming peers bootstrapped faster in comparison to standard BitTorrent. A simple comparison between the two mirrored simulations indicates that this is indeed the case.

In order to show that the faster bootstrapping (in comparison with standard BitTorrent) is sound and statistically significant we have performed a T-Test on the individual bootstrap times obtained in each run of the mirrored simulations. This test is useful for comparing the means of two distributions and showing whether possible differences in means are due to chance. A T-Test requires that the two distributions that are being compared be Normal. In our T-Test the null hypothesis (H_0) is that the mean bootstrap time of the two mechanisms are actually the same and the differences in Figure 6.2 are due to chance. In our T-Test we define a significance level of $\alpha = 0.01$. Figure 6.3 demonstrates the results of the T-Test for the First multi-swarmer. [6] The T-Test results show that H_0 is significantly rejected. It is important to note that in a limited number of cases our '*Stand_BS*' distribution fails the Normality test. Nevertheless, due to the *Central Limit Theorem* we can assume that a larger number of observations would result in a Normal distribution in these cases and that the T-Test can still be used in such cases. [7] The T-Test demonstrates that we cannot attribute the faster bootstrap time of the multi-swarmers to chance. This result supports the claim that we have made in ***H2*** and means that were not able to reject ***H2***.

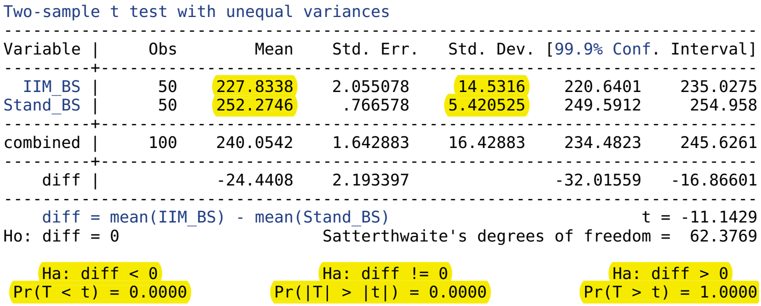
We attribute faster bootstrapping to the ability of multi-swarming peers to barter pieces of different files with each other. While in standard BitTorrent a bootstrapping leecher would have to wait for receiving an optimistic unchoke or a free piece from a seeder, a bootstrapping multi-swarmer in contrast can immediately start bartering a piece of the file that it possesses with a piece of the file that it is seeking. This implies less idle time for the multi-swarmer. Idle time is more likely to occur at the early stages of a download because some neighboring peers might not have bootstrapped. This effect can clearly be observed in Figure 6.4. In this figure we have plotted the download percentage of the first multi-swarming peer against time for the mirrored simulations. A typically observed effect that can also be seen in this figure is the downward curve for standard bootstrapping which shows that a regular peer would have to wait for some time before it can effectively utilize resource (idle time).

Note that as a result of our previous argument the change of definition for bootstrapping

[5] By expectation every peer should receive the same amount of resources as the other peers in a homogeneous environment.

[6] The T-Test results for the second multi-swarmer are similar.

[7] In cases of a failed normality test a 'non-parametric K-sample equality of means test' which does not require the distribution to be Normal reports similar results to the T-Test.



```
Two-sample t test with unequal variances
------------------------------------------------------------------------------
Variable |     Obs        Mean    Std. Err.   Std. Dev.   [99.9% Conf. Interval]
---------+--------------------------------------------------------------------
  IIM_BS |      50    227.8338    2.055078     14.5316    220.6401    235.0275
Stand_BS |      50    252.2746     .766578    5.420525    249.5912     254.958
---------+--------------------------------------------------------------------
combined |     100    240.0542    1.642883    16.42883    234.4823    245.6261
---------+--------------------------------------------------------------------
    diff |           -24.4408    2.193397               -32.01559   -16.86601
------------------------------------------------------------------------------
    diff = mean(IIM_BS) - mean(Stand_BS)                         t = -11.1429
Ho: diff = 0                            Satterthwaite's degrees of freedom =  62.3769

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 1.0000
```

Figure 6.3: Paired T-Test of ”IIM” *vs.* ”Standard BitTorrent” bootstrap time for first Multi-Swarming Peer; Difference in mean not due to chance ($\alpha = 0.01$; Pr the probability of being wrong to accept alternative hypothesis; IIM_BS (IIM Bootstrap Time); Stand_BS (Standard BitTorrent Bootstrap Time))

(see beginning of section) should have no effect on the claims that we made in ***H2*** because the notion of bootstrapping involves the ability of a peer to effectively barter pieces of the file and multi-swarmers are arguably better capable of doing so. While our simulation tool is currently only capable of simulating leecher-seeder/seeder-leecher interaction, we can still speculate that other combinations (i.e leecher-leecher/leecher-leecher) will not be outperformed by standard BitTorrent because if both multi-swarmers possess at least one piece to barter with they do not have to wait for an optimistic unchoke.

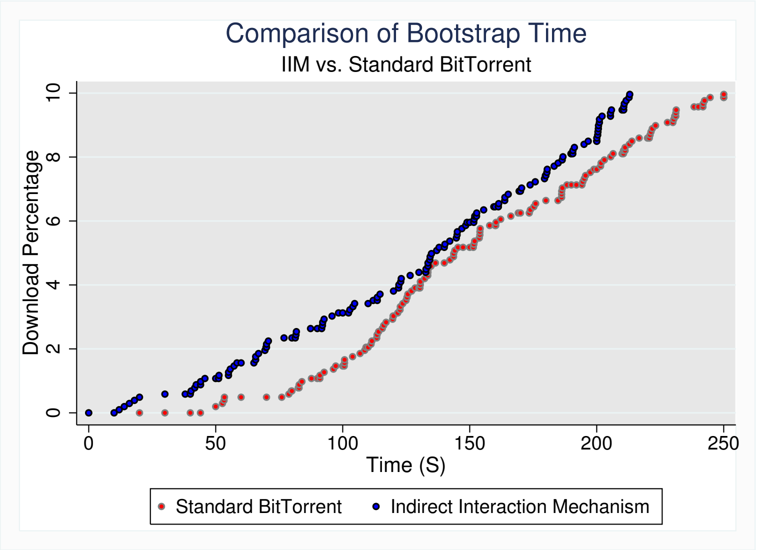


Figure 6.4: Comparison of standard BitTorrent and IIM bootstrapping for the first multi-swarming peer.

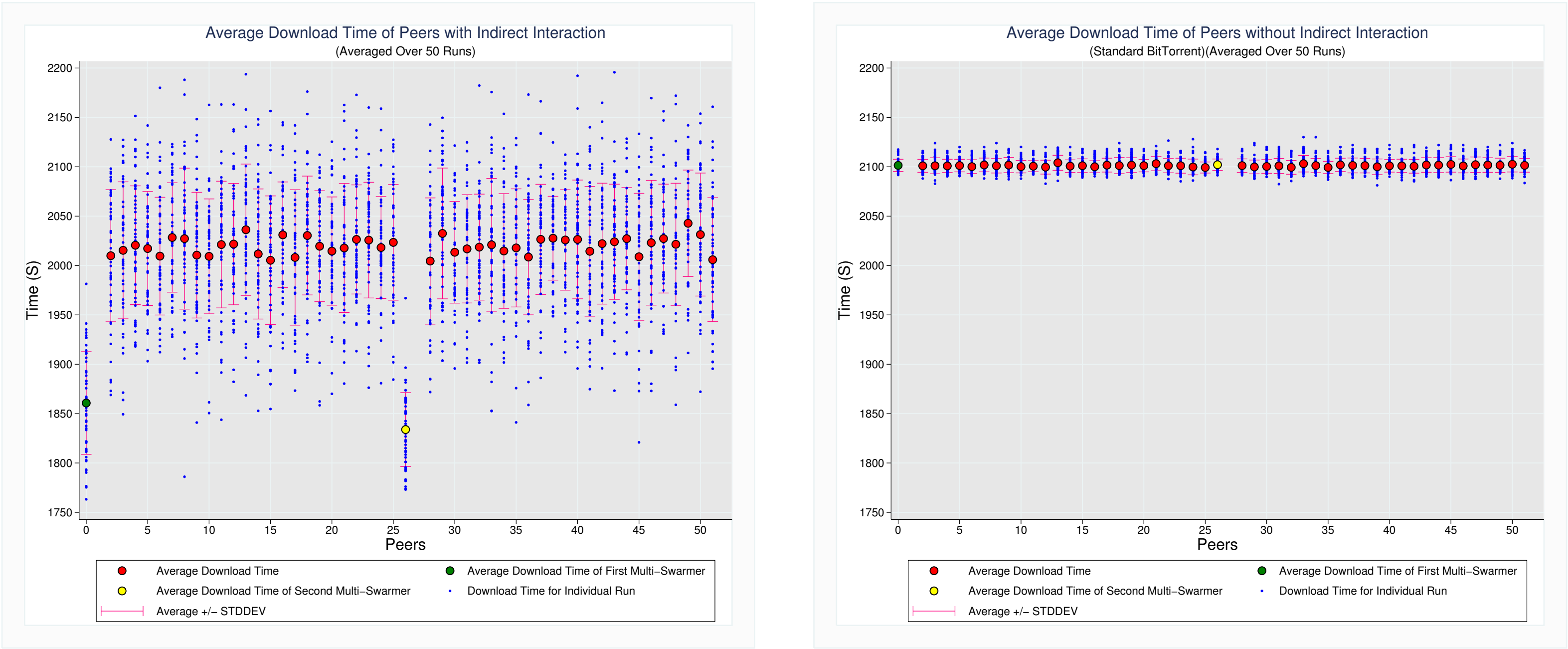
Even though idle time is a determining factor of how fast a peer will bootstrap at the beginning of a download, piece rarity is a much stronger determining factor in the later stages of a download. That is, idle time in the midst of a download is more likely to be caused by some other neighboring peer having a more rare piece of the file which will make the second peer more attractive to barter with. While we have demonstrated that multi-swarmers reduce their idle time at bootstrapping our model of IIM suggests that due to higher access to rare pieces multi-swarmers should still be able to maintain their advantage in the later stages of a download. In our next section we examine this statement by resuming the set of mirrored bootstrap simulations until all peers have finished downloading the file.

6.3 Downloading with Indirect Interaction

Our second set of simulations is directed towards attempting to reject our hypothesis regarding the download time of peers. Recall our first hypothesis (***H1***) from Section 5.1.3 in which we hypothesized that in a homogeneous environment indirectly interacting peers will download an entire file faster in comparison to standard BitTorrent. Again we speculate that a higher bargaining power in addition to more access to rare pieces (especially in leecher-seeder/seeder-leecher combinations) to be the reason. Once a peer has no option to barter in one swarm it can switch to bartering in a second swarm, thereby reducing idle time. However, this also means that a multi-swarmer has more access to rare pieces because if it does not have a rare piece of one file to provide in return for a rare piece of the same file, it is still likely to have a rare piece of the second file to barter for with the initial rare piece.

As before, we are looking for results in which the standard BitTorrent protocol outperforms IIM on average and thereby reject our hypothesis ***H1***. A negative result will also indicate that our IIM model of Section 5.1.2 is not capable of correctly predicting what would happen. Figure 6.5 demonstrates the results of simulating IIM mirrored with simulating standard BitTorrent over 50 runs. The charts are constructed as before.

Figure 6.5 demonstrates that not only are the multi-swarmers able to maintain their bootstrapping advantage, but that they were also able to increase the gap with regular leechers. As before, notice the slight decrease of the download time of regular leechers in IIM simulations in comparison to standard BitTorrent. We attribute this effect to the freed up resources that are available to the regular leechers. As before we see a low variance in the download times of the leechers in the standard BitTorrent simulations which is a result of the homogeneous setting.

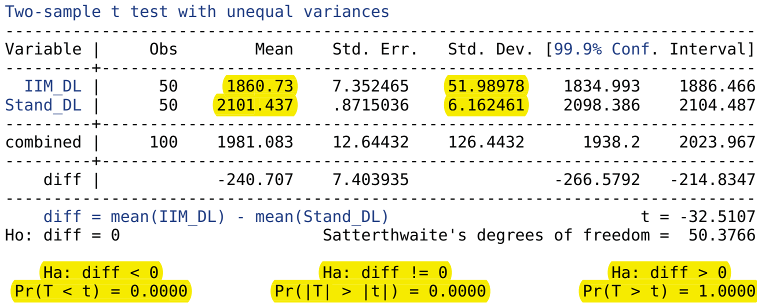


(a) Download with IIM

(b) Standard Download

Figure 6.5: Comparison of the average download time of peers in IIM and standard BitTorrent. All peers have an upload bandwidth of 512 *kbps*.

The T-Test will again help us to see if we can reject ***H1***. The null hypothesis (H_0) is that the mean download time of the two mechanisms are actually the same and the differences in Figure 6.5 are due to chance. We define a significance level of $\alpha = 0.01$. Figure 6.6 summarizes the results of the T-Test for the First multi-swarmer. [8] It can be seen that H_0 is significantly rejected. The T-Test results demonstrate that we cannot attribute the faster average download time of the multi-swarmers to chance. This result supports the claim that we have made in ***H1*** as we are not able to reject ***H1***.



```
Two-sample t test with unequal variances
------------------------------------------------------------------------------
Variable |     Obs        Mean    Std. Err.   Std. Dev.   [99.9% Conf. Interval]
---------+--------------------------------------------------------------------
  IIM_DL |      50     1860.73    7.352465    51.98978    1834.993    1886.466
 Stand_DL |      50    2101.437    .8715036    6.162461    2098.386    2104.487
---------+--------------------------------------------------------------------
combined |     100    1981.083    12.64432    126.4432      1938.2    2023.967
---------+--------------------------------------------------------------------
    diff |            -240.707    7.403935               -266.5792   -214.8347
------------------------------------------------------------------------------
    diff = mean(IIM_DL) - mean(Stand_DL)                          t = -32.5107
Ho: diff = 0                     Satterthwaite's degrees of freedom =  50.3766

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 1.0000
```

Figure 6.6: Paired T-Test of ”IIM” *vs.* ”Standard BitTorrent” download time for first Multi-Swarming Peer; Difference in mean not due to chance ($\alpha = 0.01$, Pr the probability of being wrong to accept alternative hypothesis; IIM_DL (IIM Download Time); Stand_DL (Standard BitTorrent Download Time))

We attribute faster download times to a multi-swarmers ability to bootstrap faster, it’s lowered idle time and it’s ability to download more rare pieces which will in turn make it more attractive to leechers. To demonstrate this argument more clearly note the slight drop in the average download time of regular leechers when IIM is enabled. This effect is interesting in the sense that in our current setting it would at least take 2048 seconds for the tracker to upload an entire copy of the file to the swarm. Clearly regular leechers have been able to download the file faster that this minimum time. Since the only alternatively available source for the pieces that the seeder could not have uploaded is the multi-swarming peer, it is clear that leechers should be attracted to interacting with the multi-swarming peer. For the same reasons as before we speculate that other combinations of multi-swarmers (i.e. leecher-leecher/leecher-leecher) will show similar results.

One of the reasons that in our setting multi-swarming peers are able to boost their performance is that only 1 seeder exists in the swarm and indirect interaction with a multi-swarmer resembles the exclusive services of a second seeder which regular leechers have no access to. However, once multi-swarming peers loose their advantages we expect to see a lower efficiency for IIM. For instance when the ratio of seeders to leechers in a swarm is high, many peers are able to download rare pieces of the file and therefore multi-swarmers will loose their attractiveness for regular leechers. In our next section we present a set of simulation results that examine this effect in terms of the relative download time of IIM against standard BitTorrent.

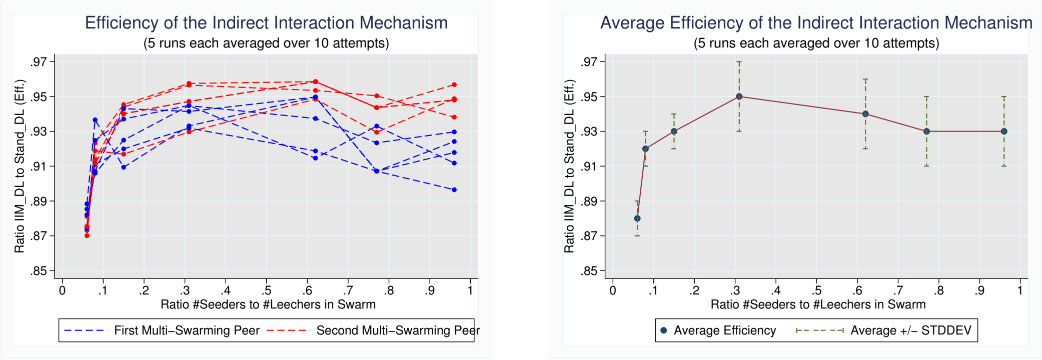
[8] The T-Test results for the second multi-swarmer are similar.

6.4 Efficiency of Indirect Interaction

When the ratio of seeders to leechers in a swarm is low, rare pieces of the file spread between peers at a slower rate than when this ratio is high. As a result multi-swarming peers that obtain such pieces through indirect interaction become more attractive to regular leechers and therefore prefer interacting with the multi-swarmers. This allows multi-swarmers to experience a faster download time. While a low seeder to leecher ratio is a characteristic of many BitTorrent swarms, this ratio can considerably vary depending on the popularity and age of the swarm.

In a series of simulation runs we have increased the number of seeders in our swarms to examine the efficiency of IIM under an increasing seeder to leecher ratio. We define the efficiency of IIM to be the download time ratio of IIM to standard BitTorrent for multi-swarmers. The results of these simulation runs are depicted in Figure 6.7. We have measured the efficiency of IIM in 7 separate simulation with $1,2,4,8,16,20$ and 25 seeders. The number of leechers remain the same as the original setting. At a $\frac{1}{26}$ ratio of seeders to leechers file pieces are rare and leechers would have to compete to download the file. On the other hand, a $\frac{25}{26}$ ratio of seeders to leechers means that file pieces are practically for free and almost all leechers can enjoy an increase in download speeds almost equal to the upload speed of a seeder which comes without the need for the leechers to reciprocate.

As expected, IIM is most efficient when the seeder/leecher ratio is low. In fact, it turns out that if a swarm has zero seeders, at least one of the multi-swarmers is still able to download the entire file under our current setting. However, IIM still maintains an advantage of better download times even when the seeder/leecher ratio is high. We can attribute this effect to the fair allocation of upload slots by seeders. That is, no single peer, including non of the multi-swarmers, can exclusively enjoy the free services of a seeder. According to the BitTorrent protocol seeders assign upload slots to leechers in a round-robin fashion when all leechers download at the same speed. As a result file pieces become less rare, however, multi-swarmers still remain more attractive because they also enjoy the services of many seeders. As demonstrated in Figure 6.7 IIM has been at least 5% more efficient



(a) (b)

Figure 6.7: Efficiency of IIM against Standard BitTorrent.

than standard BitTorrent in our simulations while being 11% more efficient in best case.

Up to this point we have demonstrated simulation results that were obtained from a homogeneous environment in which all peers have the same upload and download speed. We have observed results that were in line with our analysis of the IIM model in this environment. However, in practice BitTorrent peers tend to interact in a heterogeneous environment with multiple classes of upload/download speeds. As explained in Chapter 5 it is more difficult to derive coherent hypotheses in a heterogeneous environment from our IIM model. Simulations are probably a better way of demonstrating properties of the mechanism in a heterogeneous environment . The following section is dedicated to presenting a limited number of simulations of IIM in such a setting.

6.5 Simulating a Heterogeneous Environment

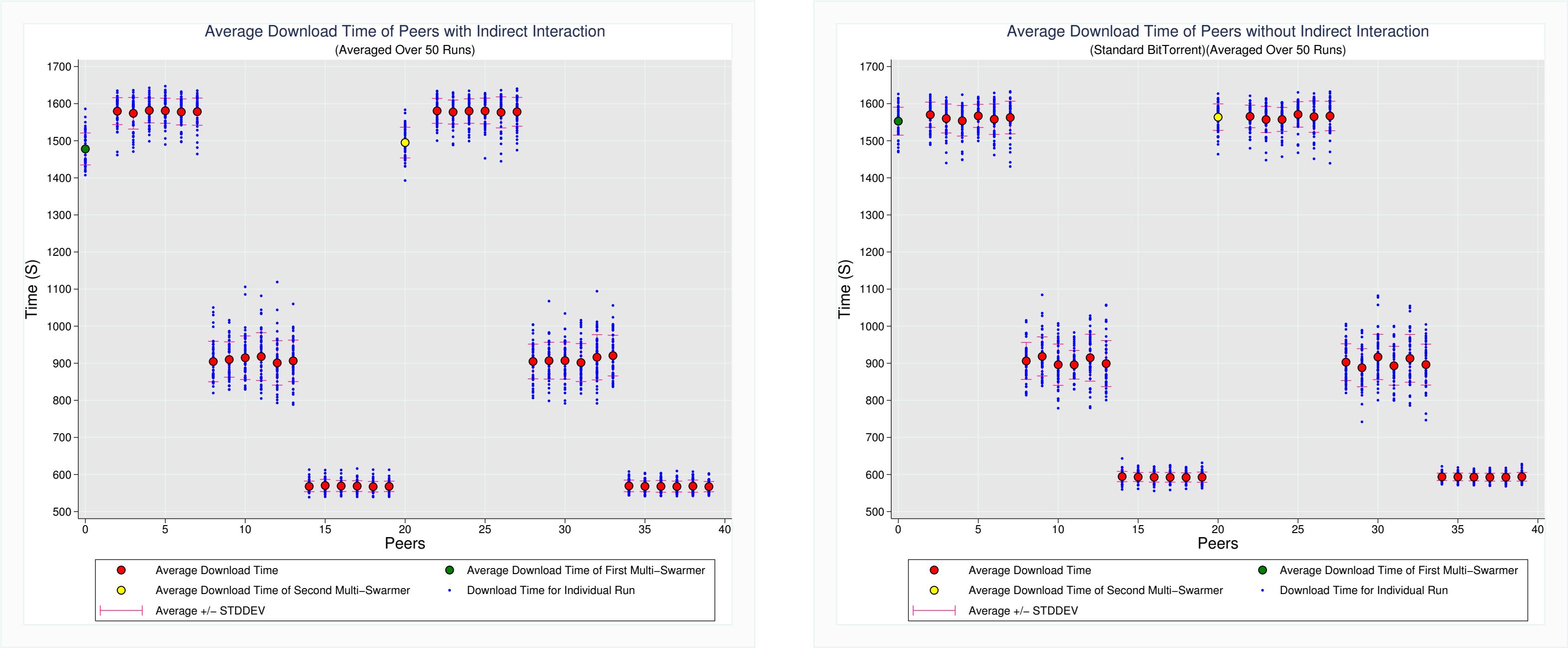
In our simulation of a heterogeneous environment we have modified the upload speeds of the seeders/leechers and their numbers according to table 6.1. We have created three different classes of peers with 512, 1024 and 2048 *kbps* of upload speed. Each swarm contains 6 leechers of each class. Each swarm has a single seeder with 2048 *kbps* of upload speed.

Table 6.1: Setting of each swarm.

#Peers	**Upload Speed Class**
6 leechers + 1 seeder	2048 *kbps*
6 leechers	1024 *kbps*
6 leechers	512 *kbps*

We conduct four sets of simulations in which the two multi-swarming peers are firstly assigned the same upload speed of 512 *kbps*, secondly assigned different upload speeds of 512 and 1024 *kbps*, thirdly assigned different upload speeds of 512 *kbps* and 2048 *kbps* and finally upload speeds of 1024 and 2048 *kbps*. In our second, third and fourth sets of experiments the multi-swarmers have been modified to prefer interacting with each other over other peers. The reason is that under standard BitTorrent such interaction would not occur because the unchoking algorithm would choose peers of a higher upload bandwidth to unchoke.

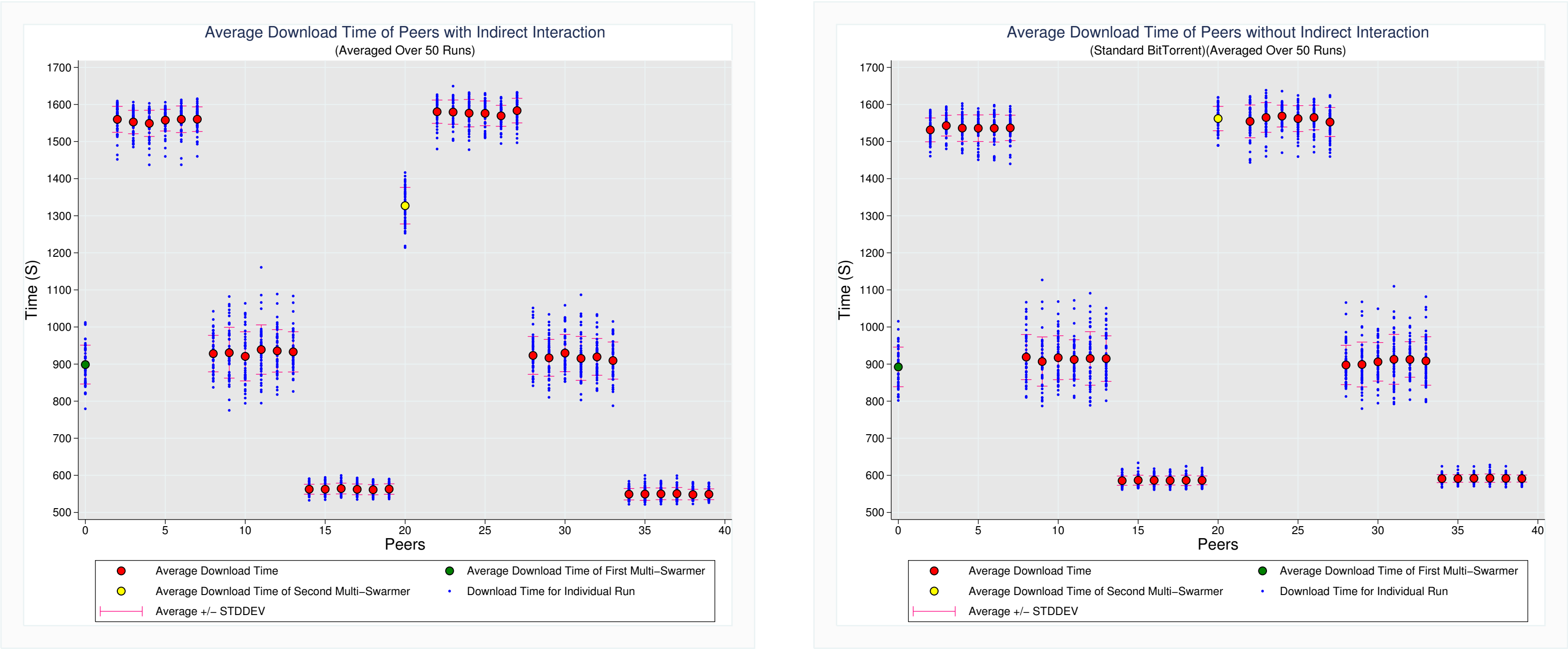
The first, second, third and fourth simulation results are respectively demonstrated in Figures 6.8, 6.9, 6.10 and 6.11. The classification of peers is evident in all four figures as peers that have a higher upload speed download the file faster. This effect is attributed to a clustering phenomenon in which peers of a class tend to unchoke peers of the same or higher classes more than other classes of peers [28].



(a) Download with IIM

(b) Standard Download

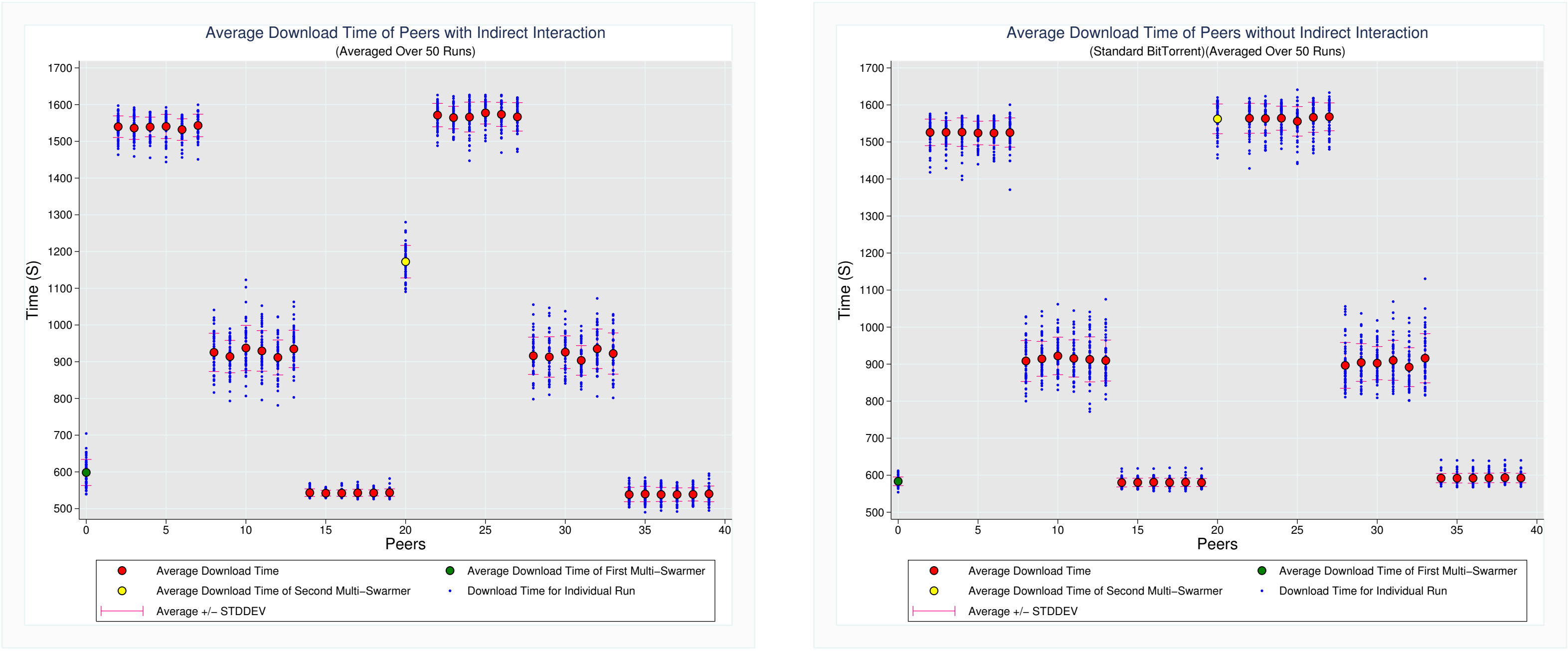
Figure 6.8: Comparison of the average download time of peers in IIM and standard BitTorrent in a Heterogeneous Environment; Both Multi-Swarmers have an upload speed of 512 *kbps*; Peers 1 and 21 seeders with 2048 *kbps* upload; peers 2-7 and 22-27 leechers with 512 *kbps* upload; peers 7-13 and 28-33 leechers with 1024 *kbps* upload speed; peers 14-19 and 34-39 leechers with 2048 *kbps* upload.



(a) Download with IIM

(b) Standard Download

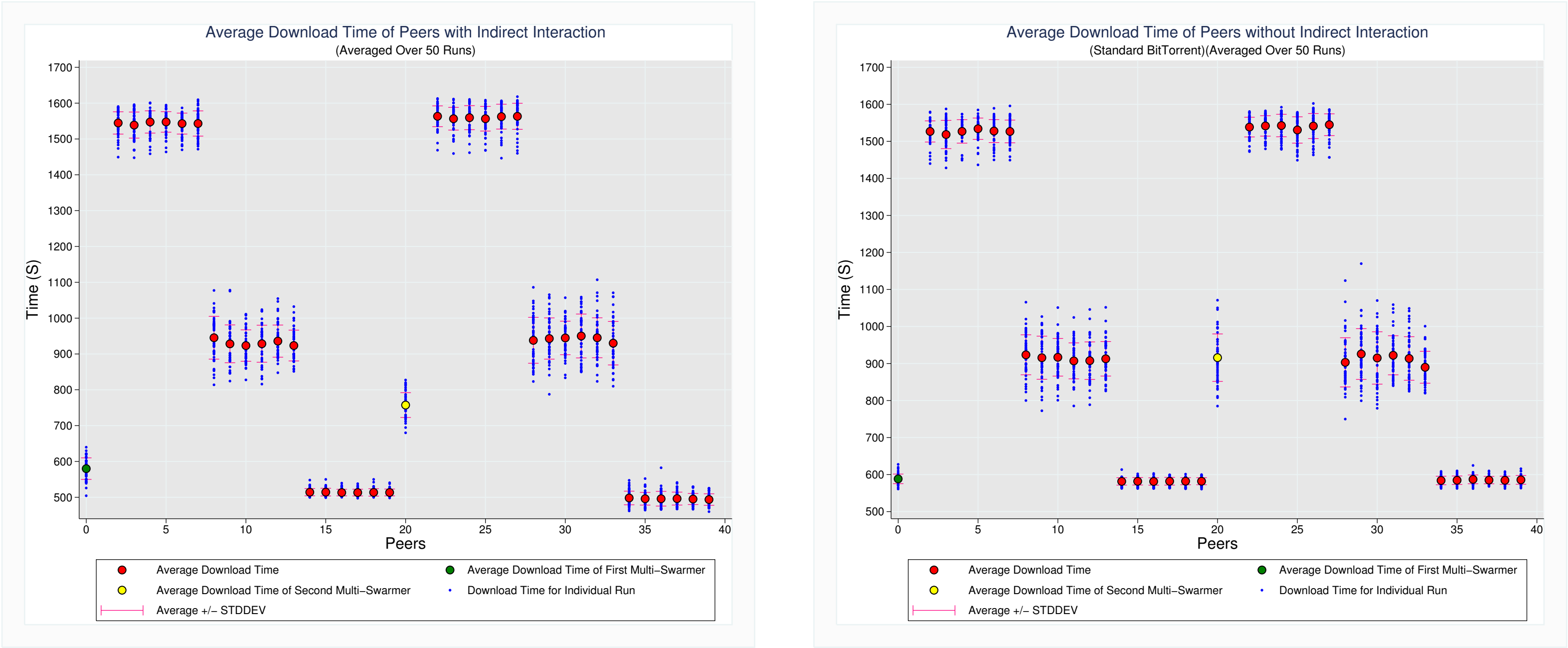
Figure 6.9: Comparison of the average download time of peers in IIM and standard BitTorrent in a Heterogeneous Environment; First Multi-Swarming peer 1024 *kbps* upload; Second multi-swarming peer 512 *kbps* upload; Peers 1 and 21 seeders with 2048 *kbps* upload; peers 2-7 and 22-27 leechers with 512 *kbps* upload; peers 7-13 and 28-33 leechers with 1024 *kbps* upload speed; peers 14-19 and 34-39 leechers with 2048 *kbps* upload.



(a) Download with IIM

(b) Standard Download

Figure 6.10: Comparison of the average download time of peers in IIM and standard BitTorrent in a Heterogeneous Environment; First Multi-Swarming peer 2048 *kbps* upload; Second multi-swarming peer 512 *kbps* upload; Peers 1 and 21 seeders with 2048 *kbps* upload; peers 2-7 and 22-27 leechers with 512 *kbps* upload; peers 7-13 and 28-33 leechers with 1024 *kbps* upload speed; peers 14-19 and 34-39 leechers with 2048 *kbps* upload.



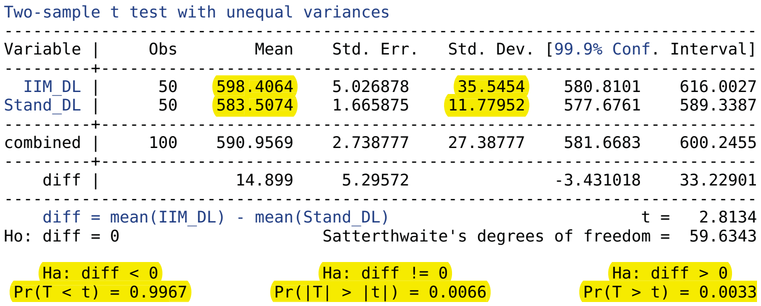
(a) Download with IIM

(b) Standard Download

Figure 6.11: Comparison of the average download time of peers in IIM and standard BitTorrent in a Heterogeneous Environment; First Multi-Swarming peer 2048 *kbps* upload; Second multi-swarming peer 1024 *kbps* upload; Peers 1 and 21 seeders with 2048 *kbps* upload; peers 2-7 and 22-27 leechers with 512 *kbps* upload; peers 7-13 and 28-33 leechers with 1024 *kbps* upload speed; peers 14-19 and 34-39 leechers with 2048 *kbps* upload.

When both multi-swarmers belong to the same class of peers (Figure 6.8) we observe that indirect interaction leads to a faster download time for both multi-swarmers as before. [9] Just as before we attribute this effect to their attractiveness to other leechers (mostly from their own class). On the other hand when peers belong to different classes of peers (Figures 6.9, 6.10 and 6.11) the slower multi-swarmer visibly benefits from the interaction while the benefit for the second multi-swarmer is in question especially when the difference in bandwidth is larger (Figure 6.10). Because we have modified the multi-swarmers to prefer interacting with each other in the second simulation, the faster peer is effectively choosing to download a rare piece instead of a less rare piece that can be downloaded at a higher speed. In standard BitTorrent the choice in this trade off would have been to download the less rare piece at a higher speed. Such a trade off could affect the overall download time of the peer.

It turns out that the trade off between rarity and download speed does indeed effect the overall download time of the faster multi-swarming peer. Figure 6.12 depicts the T-Test results of comparing the IIM download time of the faster multi-swarmer with its download time in standard BitTorrent for the simulations depicted in Figure 6.10. The null hypothesis (h_0) is that the two distributions have the same mean. As seen in Figure 6.12 the null hypothesis can be rejected in favor of the hypothesis that standard BitTorrent download time is less than the IIM download time for this instance.



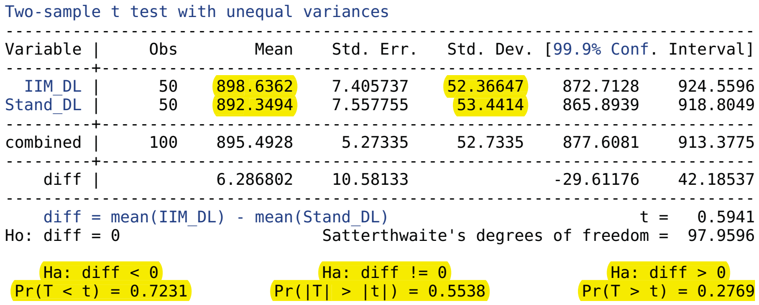
```
Two-sample t test with unequal variances
------------------------------------------------------------------------------
Variable |     Obs        Mean    Std. Err.   Std. Dev.   [99.9% Conf. Interval]
---------+--------------------------------------------------------------------
  IIM_DL |      50    598.4064    5.026878     35.5454    580.8101    616.0027
Stand_DL |      50    583.5074    1.665875    11.77952    577.6761    589.3387
---------+--------------------------------------------------------------------
combined |     100    590.9569    2.738777    27.38777    581.6683    600.2455
---------+--------------------------------------------------------------------
    diff |              14.899     5.29572               -3.431018    33.22901
------------------------------------------------------------------------------
    diff = mean(IIM_DL) - mean(Stand_DL)                          t =   2.8134
Ho: diff = 0                          Satterthwaite's degrees of freedom =  59.6343

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.9967         Pr(|T| > |t|) = 0.0066          Pr(T > t) = 0.0033
```

Figure 6.12: Paired T-Test of "IIM" *vs.* "Standard BitTorrent" download time for first Multi-Swarming Peer in Figure 6.10; standard BitTorrent download time faster ($\alpha = 0.01$, Pr the probability of being wrong to accept alternative hypothesis; IIM_DL (IIM Download Time); Stand_DL (Standard BitTorrent Download Time))

On the other hand a T-Test comparison of download times for the simulations depicted in Figures 6.9 and 6.11 show a different result. In both of these two cases the bandwidth difference of the two multi-swarming peers is not as much as the previous case. Figures 6.13 and 6.14 depict the T-Test results for these two cases. None of the results are able to significantly reject the null hypothesis. However, for the case of multi-swarming peers having bandwidths of 1024 and 2048 *kbps* (belonging to the faster classes of peers) results leans towards a hypothesis that IIM has outperformed standard BitTorrent. For the other case in which the multi-swarming peers have bandwidths of 512 and 1024 *kbps* results

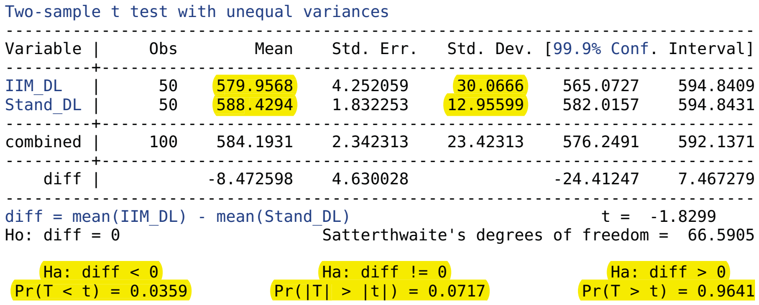
[9] T-Test results significantly reject the null hypothesis.

could lean in either direction. As the null hypothesis cannot be significantly rejected we could state that there is no difference in the performance of BitTorrent and IIM in this particular case.



```
Two-sample t test with unequal variances
------------------------------------------------------------------------------
Variable |     Obs        Mean    Std. Err.   Std. Dev.   [99.9% Conf. Interval]
---------+--------------------------------------------------------------------
  IIM_DL |      50    898.6362    7.405737    52.36647    872.7128    924.5596
Stand_DL |      50    892.3494    7.557755     53.4414    865.8939    918.8049
---------+--------------------------------------------------------------------
combined |     100    895.4928     5.27335     52.7335    877.6081    913.3775
---------+--------------------------------------------------------------------
    diff |            6.286802    10.58133               -29.61176    42.18537
------------------------------------------------------------------------------
    diff = mean(IIM_DL) - mean(Stand_DL)                          t =   0.5941
Ho: diff = 0                     Satterthwaite's degrees of freedom =  97.9596

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.7231         Pr(|T| > |t|) = 0.5538          Pr(T > t) = 0.2769
```

Figure 6.13: Paired T-Test of "IIM" *vs.* "Standard BitTorrent" download time for first Multi-Swarming Peer in Figure 6.9; Standard BitTorrent Download Time faster however null hypothesis cannot be rejected ($\alpha = 0.01$, Pr the probability of being wrong to accept alternative hypothesis; IIM_DL (IIM Download Time); Stand_DL (Standard BitTorrent Download Time))



```
Two-sample t test with unequal variances
------------------------------------------------------------------------------
Variable |     Obs        Mean    Std. Err.   Std. Dev.   [99.9% Conf. Interval]
---------+--------------------------------------------------------------------
IIM_DL   |      50    579.9568    4.252059     30.0666    565.0727    594.8409
Stand_DL |      50    588.4294    1.832253    12.95599    582.0157    594.8431
---------+--------------------------------------------------------------------
combined |     100    584.1931    2.342313    23.42313    576.2491    592.1371
---------+--------------------------------------------------------------------
    diff |           -8.472598    4.630028               -24.41247    7.467279
------------------------------------------------------------------------------
diff = mean(IIM_DL) - mean(Stand_DL)                          t =  -1.8299
Ho: diff = 0                     Satterthwaite's degrees of freedom =  66.5905

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.0359         Pr(|T| > |t|) = 0.0717          Pr(T > t) = 0.9641
```

Figure 6.14: Paired T-Test of "IIM" *vs.* "Standard BitTorrent" download time for first Multi-Swarming Peer in Figure 6.11; IIM Download Time faster however null hypothesis cannot be rejected ($\alpha = 0.01$, Pr the probability of being wrong to accept alternative hypothesis; IIM_DL (IIM Download Time); Stand_DL (Standard BitTorrent Download Time))

To sum up, the heterogeneous simulation results show that when multi-swarmers belong to the same class of peers the results are similar to those observed in a homogeneous setting and IIM outperforms BitTorrent. However when these peers belong to different classes a trade off is involved. When the bandwidth difference is too large BitTorrent performs better and when the bandwidth difference is little IIM performs better or no worse than BitTorrent. This means that IIM should still incentivize multi-swarming peers to interact indirectly as long as the bandwidth difference is not too large.

At this point we have presented a summary of the experiments that we have conducted to examine the properties of the Indirect Interaction Mechanism. In our next section we focus on the conclusions that can be drawn from the demonstrated results and indicate possibly interesting areas for further experimentation.

6.6 Conclusions

In this chapter we have simulated the Indirect Interaction Mechanism within a homogeneous and a heterogeneous setting. The simulations are structured in a way that allow us to empirically investigate the properties of the mechanism. Moreover, the simulation results allow us to verify our game theoretical model of the mechanism (Chapter 5) in addition to justifying the hypotheses that were derived from the model.

Our simulations in the homogeneous setting show that multi-swarming peers do indeed experience faster downloading and faster bootstrapping. These phenomena correspond to our first and second hypotheses from Section 5.1.3 respectively. In addition T-Tests (and non-parametric comparisons of means) reject the hypothesis that these advantages are the result of chance. Moreover, our simulations also show that IIM is able to maintain a reasonable efficiency in comparison with standard BitTorrent.

On the other hand our simulations in the heterogeneous environment which better match the real state of the world show positive results as well. That is, we can conclude that as long as indirect interaction takes place between peers that do not vary in upload speed IIM performs better or no worse than standard BitTorrent. We can also conclude that IIM has a limited tolerance to indirect interaction between peers that have different upload bandwidths. That is, peers with different upload bandwidths could still be able to interact indirectly while both retaining the faster download times. Further experimentation in this setting should allow us to gain a better understanding of how much tolerance IIM can exhibit and find limits for the differences in upload speed that should result in performance boost and differences that would lead to opposite outcomes.

While we have been able to confirm most of our hypotheses in this chapter a third hypothesis which involves the indirect interaction of other combinations of multi-swarmers (i.e leecher-leecher/leecher-leecher) still remains unconfirmed. In order to conduct experiments that test the validity of this third hypothesis we require a more sophisticated simulation tool than the one that we have developed. In the future we plan to extend our current simulation tool to be able to conduct experiments with other combinations of multi-swarmers. At the same time there are many interesting questions that we like to find an answer to.

The first interesting question that we like to find an answer is how the properties of IIM would change if we take the properties of the underlying network (e.g. network delays) into account. Accounting for network properties in our model and in our simulation would get us even closer to the state of the real world. Ultimately we would like to implement the Indirect Interaction Mechanism completely and use real world simulations to examine its properties. It could be that the efficiency of IIM is decreased as a result of network delays.

A second question that we like to approach is whether the effects of indirect interaction are cumulative. That is, could the efficiency of IIM increase as the result of a multi-swarming peer being able to interact with more than just one other multi-swarmer. We can only speculate this to be true based on our understanding of our model of IIM. We would also like to extend our simulation for larger cyclic indirect interaction that just a cycle of length 2. It is likely that unaccounted properties such as network delays play a crucial role in an extended experimentation setting.

A third interesting question is the effect of IIM on the traditional notion of seeding. Given the advantages of indirect interaction for multi-swarmers that can seed one of their files (i.e. leecher-seeder) We would like to find out if such advantages would cause the population of peers to abandon the traditional notion of seeding for indirect interaction, which is a notion of conditional seeding, or whether both notions could co-exist from an evolutionary perspective.

A fourth and final question that we like to find an answer to pertains to the possible ways of exploiting the mechanism. While we have covered many of the properties of IIM (including some limited unexploitability characteristics) we have mostly assumed that peers follow the standard BitTorrent protocol. It is interesting to know if and how IIM is exploitable and what the impact of the exploits would be on the properties of the mechanism.

Chapter 7

Conclusions and Future Work

In achieving our research goal, "Incentivizing Seeding in BitTorrent", we have had to design a centralized mechanism that is able to match the performance of the BitTorrent protocol while at the same time avoiding the complexities and pitfalls of current reputation systems (or monetary mechanisms) such as exploitability.[1] *In addition our aim is for the mechanism to be scalable and/or alternatively easily implementable as a (more scalable) distributed mechanism given currently existing technologies. Finally, we also aim to leverage a large body of work in the closely related fields of experimental economics and social psychology to determine if our design is in line with recent findings regarding the human psyche and how users would react when using the proposed mechanism. In this chapter we look back at our findings and contemplate on how well our research goals have been achieved.*

This chapter is organized as follows. We will fist summarize our work and our findings with respect to each chapter of work in Section 7.1. We will then contemplate on our findings and provide our concluding remarks with respect to each of our design goals in Section 7.2. Our conclusions without doubt will be the subject of future debate and we will reflect on the advantages and disadvantages of implementing our proposed mechanism in Section 7.3. We end this chapter with our recommendations for future work in Section 7.4.

7.1 Summary

Let us start by re-iterating our research goal In order to see how much of our initial goals we have achieved. We defined our research goal to be as follows:

> Our research goal is to design an alternative mechanism of incentivizing seeding in BitTorrent to the current reputation systems or sharing ratio enforcement methods. Our aim is avoid the pitfalls of such mechanisms such as exploitability and at the same time maintain BitTorrent's performance. Additionally, we want the mechanism to benefit all peers. Finally, we want the mechanism to be

[1] Refer to Chapter 2 for an overview of the pitfalls of reputation systems that are currently in use in BitTorrent.

scalable and/or easily implementable as a distributed mechanism with higher scalability.

We start with a key observation of the difference between BitTorrent and previous generations of P2P file sharing systems. The difference is that unlike previous P2P systems that are based on peers exchanging different files with each other, BitTorrent is based on peers exchanging pieces of a single file with each other. This difference is largely due to the implementation of a TFT mechanism in the BitTorrent protocol which is attributed to be one of its main success factors. However, peers that already possess an entire file cannot exchange pieces of this file for pieces of other files and only give away the file for free. Clearly this can be attributed as one of the possible reasons behind a lack of incentive to seed. Therefore, we have speculated that BitTorrent peers in general are missing out on an opportunity to barter pieces of different files (indirect interaction).

From this initial speculation we begin our main body of work which can be summarized into three main steps as demonstrated in Table 7.1. We summarize our work and findings in each step in what follows.

Table 7.1: Summary of work.

Step	**Description**	**Corresponding Chapter**
1	Conduct a measurement study of a private BitTorrent community (the FileList.org tracker)	3
2	Design a centralized mechanism for incentivizing seeding based on the measurement study	4
3	Analyze and simulate the mechanism to demonstrate that it achieves the main goal	5, 6

(**Step 1**) In order to investigate our initial speculation we have had to conduct a measurement study to show that indirect interaction is possible between peers that are active in multiple swarms (*multi-swarming* peers) [2]. According to our measurement study of the trace logs of the FileList.org tracker we observe that indirect interaction in the form of closed cycles of length 2 is possible with a probability of more than 80% over a period of an entire month (Section 3.4). Closed loops of indirect interaction are of specific interest because they are in essence an extension of the TFT mechanism in BitTorrent. Interestingly, the probability for indirect interaction approaches 100% when cycles of length 3 are considered (Section 3.4). Additionally, we observe that these figures are also significantly high among a small random sample of swarms from the tracker (Section 3.4). We have argued that our measurement results should not be influenced by our specific choice of studying a private tracker which differs from public trackers (i.e. sharing ratio enforcement is used in FileList.org). Finally, we reason about why the results are also applicable for public BitTorrent trackers (see Section 3.4.3). Our measurements allow us to confirm that our initial speculation is correct and that there is a large probability that BitTorrent peers could exchange pieces of multiple files.

[2] Refer to Chapter 3 for our definitions of indirect interaction and multi-swarming.

(**Step 2**) As a consequence of our findings in Step 1 we have designed a centralized mechanism which we refer to as the Indirect Interaction Mechanism (IIM). IIM leverages the opportunity of indirect interaction between (would be) seeders that are leeching at the same time (see Section 4.2). This will allow them to barter pieces of the file(s) that they (can) seed for pieces of the file(s) that they leech. IIM is designed in a way that other multi-swarmers are also able to interact indirectly (i.e. peers that leech multiple files). Nevertheless our main goal is to facilitate the former type of interaction. We propose a method of implementing IIM over multiple trackers (Section 4.3). In addition we have demonstrated that IIM is easily implementable as a distributed mechanism in combination with epidemic protocols such as BuddyCast [44] and theoretically possible to implement within the framework of DHTs (Section 4.5). In a simulation of the FileList.org tracker (with real data obtained from the traces) we have observed that IIM scales linearly with the number of multi-swarmers seeking indirect interaction (Section 4.2). We also observe that it would take the tracker an average of $10ms$ to respond to a request from a multi-swarmer in worst case. The worst case scenario being a simultaneous request for indirect interaction by all multi-swarming peers at the same moment. Based on these results we can conclude that IIM is feasible to implement. Finally we demonstrate that IIM is immune to some of the exploits that exist in reputation systems (Section 4.4).

(**Step 3**) Our third and final step involves demonstrating that IIM achieves our goal of incentivizing seeding. In order to demonstrate this property we model (Chapter 5) and simulate (Chapter 6) IIM to study its properties. We model IIM in two ways: **i**) *A game theoretical model* **ii**) *An experimental economics game with humans*. Our game theoretical model is suitable for modeling both the standard BitTorrent protocol and our proposed mechanism (Sections 5.1.1 and 5.1.2). Our analysis of the game theoretical model leads to the hypotheses that IIM enables a faster download (**H1**) and a faster bootstrapping time (**H2**) in a homogeneous environment (Section 5.1.3). A third hypothesis is left to be verified in the future and states that IIM should not be significantly outperformed by standard BitTorrent in a homogeneous environment (**H3**). While our game theoretical model allows us to analyze what the rational choices in a game theoretic setting should be, we have also considered an experimental economics approach. We argue that BitTorrent can be modeled as a *Public Goods* game by definition (Section 5.2.2). Furthermore, we argue that the *Public Goods / Indirect Reciprocity* game [35, 40, 48] closely resembles our IIM mechanism and is as close as we can get to modeling the mechanism as an experiment with human subjects given our current knowledge of the former field of study and existing experimental studies (Section 5.2.3). We have used the results of this particular experiment to analyze human behavior with respect to IIM despite the fact that one of the key requirements of this setting cannot be met. The missing requirement is a transparency of all the actions of the players which can only partially be accounted for in IIM. We can only conclude that people should show indications of higher willingness to cooperate in the game which translates to seeding files in BitTorrent. Nevertheless the willingness to seed could fade away as a result of the lack of transparency based on the experimental model (Section 5.2.4).

Finally we conduct various experiments by simulating IIM in a homogeneous and heterogeneous setting. Through our simulations we have examined our hypotheses and the validity of our game theoretical model of IIM. We have observed that IIM does indeed lead to a faster download and bootstrap time in a homogeneous setting (Sections 6.2 and 6.3). We have supported our hypotheses through additional statistical methods (T-Test) by comparing the mean download/bootstrap time of peers in IIM with peers in Standard BitTorrent. The statistics show a significant performance boost. We have also analyzed other properties of IIM such as its efficiency in comparison with standard BitTorrent and observed that IIM maintains a 5% performance boost in worst case (Section 6.4). In addition to our previous simulations we have conducted simulations in a heterogeneous environment. We observe that multi-swarmers of similar bandwidth gain the same benefits in a heterogeneous setting. Additionally we observe that in a heterogeneous setting the IIM mechanism closely matches the performance of standard BitTorrent when the difference in the bandwidths of indirectly interacting peers is not too large. This is especially true when the peers belong to the faster classes of peers (Section 6.5). Given our game theoretical model and hypotheses from Chapter 5 as a backing theory in addition to our simulation and statistical test results from Chapter 6 we can conclude that our game theoretical model of IIM is a suitable representation of IIM (and standard BitTorrent). We can also conclude that our hypotheses are valid (cannot be rejected) as they are supported by the simulation and statistical test results. The faster download and bootstrap times lead us to conclude that IIM incentivizes seeding because it leads to a performance boost for multi-swarming peers in most cases.

Having summarized our findings we move on to our conclusions with respect to each of our research goals.

7.2 Contemplating on the Goals

With our results we can conclude that the Indirect Interaction Mechanism achieves most of our research goals. Specifically we have designed a mechanism that incentivizes seeding of files because it provides multi-swarmers with a performance boost. IIM is not limited to private communities and is implementable on public trackers. All peers that have a correctly configured firewall are able to benefit from IIM on a public tracker. IIM requires less message passing [3] between peers and other entities in the BitTorrent protocol such as the tracker which leads to a lower complexity. Additionally, IIM does not suffer from Sybil exploits which most reputation systems suffer from because it does not allow earned credit to be transfered from one transaction to another. Moreover, IIM is immune to phenomena such as the BitCrunch [24] that allow higher bandwidth peers to gain exceptionally high reputations and in effect discriminate against lower bandwidth peers. Despite all IIM's achievements, it scales linearly with the number of peers using the mechanism. Additionally, IIM can be easily hooked up with existing epidemic protocols such as BuddyCast to operate as a decentralized mechanism with better scalability.

We can summarize our conclusions with respect to achieving our research goals as depicted in Table 7.2. In our next section we move on to considering the pros and cons of implementing our mechanism.

[3] This is both in terms of the size of the message and the number of messages

Table 7.2: Summary of research goals and achievement status

Goal	Status	Additional Notes
Incentivize Seeding	Achieved	Conditional seeding has been achieved and there is a high probability for it to occur
Maintain Performance	Achieved	Bootstrap and download times improved.
Exploit Proof	Moderately Achieved	IIM is Sybil-proof, Incentive Compatible and DDoS-proof. Large view exploit still possible.
Fairness	Achieved	Peers of all bandwidths are able to benefit and there is no discrimination
Available to the Public	Achieved	All peers that have a correctly configured firewall can benefit
Scalability	Moderately Achieved	The mechanism is feasible to implement and scales linearly (polynomially in other cases)
Decentralization	Achieved	The mechanism can be extended to operate as a fully distributed mechanism in combination with existing technologies

7.3 Discussion

In this section we concentrate on the implications of our research an the advantages and disadvantages for implementing IIM in practice. We present reasons that are in favor of IIM followed by reasons that are against IIM:

- **Advantages**
  - *Reduced Complexity*: In terms of complexity, IIM requires very little communication between peers and the tracker in comparison with existing mechanisms that incentivize seeding. That is peers require the tracker to find possible opportunities of indirect interaction at the same frequency with which they contact the tracker to get a list of peers in standard BitTorrent. Moreover, a single indirect interaction request to the tracker is likely to be enough for the whole duration of a download. In addition peers require only a constant number of messages to be passed between them to start indirect interaction. On the other hand reputation systems and sharing ratio enforcement mechanisms require regular message passing to indicate how many bytes they have uploaded/downloaded. As long as the frequency of message passing in such mechanisms is higher than that of the request messages to a tracker, IIM is superior in terms of the number of messages passed. Moreover, reputation systems require multiple messages to be passed regularly to all neighbors of a peer in order to allow the computation of a reputation score. As a result the number of messages can easily grow as the number of neighbors grow.
  - *Unexploitability*: Most of the existing reputation systems are susceptible to Sybil attacks in which a single peer impersonates multiple identities and fakes

transactions with peers that are actually impersonations of itself to gain higher reputation. Other reputations systems are vulnerable to peers just lying about their reputation scores. [4] Nevertheless, most of such vulnerabilities occur because peers are able to accumulate earned credits and use them elsewhere. IIM in contrast does not allow credit to accumulate or to be transfered. That is, indirectly interacting peers can be viewed as giving each other credit that is not cumulative and has to be spent immediately at the same place as it was issued. Therefore, a peer that impersonates multiple identities is not able to transfer false credit that it earns by interacting with one of its own sybils to other peers. Finally no peer can simply lie because each of the involved parties can verify the correctness of information and at the same time the mechanism is designed in a way that lying about information does not give an untruthful peer any advantages. Even though exploiting reputation and sharing ratio enforcement mechanism requires a detailed knowledge of how these mechanism work and have mostly been demonstrated in lab experiments it is safer to design mechanisms that are immune to such vulnerabilities even if they are rarely observed in the real world.

  - *Fairness*: IIM is a fair mechanism because peers of all bandwidth capabilities can benefit provided they find other peers that have a relatively close bandwidth to them. In contrast reputation systems and sharing ratio enforcement mechanisms lead to a discriminatory phenomena in which high bandwidth peers are able gain high scores. In turn, this allows them to attract a large portion of the peers while low bandwidth peers have to wait for long periods to attract peers or gain enough credit to allow them to interact with others. This phenomena is unfair because low bandwidth peers that are following protocol and devoting their full resources as BitTorrent requires them to are punished along side peers that withhold resources or freeride.
  - *Convenience for the Peers*: IIM is a very convenient mechanism for the users. The convenience comes from the fact that users do not have to stay connected to gain credit or reputation. A user can merely start multi-swarming at any time and immediately benefit from indirect interaction without having to wait for a high reputation to benefit. In contrast most reputation systems and sharing ratio enforcement mechanism require users to stay connected for long periods in order to gain or maintain a high status.

- **Disadvantages**
  - *Lack of Scalability*: IIM does not scale well with a growing number of users. Even though IIM scales linearly (or polynomially depending on the configuration) it is one of its main weaknesses. Our figures suggest that IIM is feasibly implementable on a tracker like FileList, which was the subject of our measurement study. Nevertheless popular trackers like *the piratebay* are much larger in

[4] There are some known techniques for limiting the effect of this vulnerability. See EigenTrust for an example [26].

terms of the number of swarms and numbers of peers. This means that unless IIM is configured to be limited to a certain number of swarms on such popular trackers it could be infeasible to implement. Limited configurations could be based upon strategies such as consideration of only popular swarms, consideration of relatively new swarms, etc. to make IIM feasible.

- *Central Point of Failure*: One of the main disadvantages of IIM which is a direct consequence of its centralized design is that it is dependent on a single entity which can fail. This creates a single point of failure that can harm IIM's performance.
- *Changing Protocol Specifications*: In order to successfully implement IIM, many BitTorrent clients have to modify their code and adopt the IIM protocol specification. Another alternative is for a single popular client to implement IIM. Either way a critical mass of users is required to make IIM successful. It could take a considerable amount of time to reach a critical mass of users.
- *Privacy*: IIM is dependent on peers revealing information to each other that could be considered as private information. Therefore IIM faces a privacy issue. This issue could be handled with a simple user license agreement. However, If such a solution is unacceptable based on privacy laws modifications to the protocol are required to protect user privacy.

Having explained the benefits and disadvantages of IIM we end this chapter with our recommendations for future work.

7.4 Recommendations for Future Work

Throughout our work we have faced some limitations and have been unable to answer some important questions. For example in our measurement study of the FileList.org tracker we have been limited by the granularity of our data set. As a result of how the data set has been compiled we are limited to having information on the activity of each peer with gaps of several minutes. The granularity of the data has lead us to assume that peers remain online during the gaps in our trace. Furthermore, we have been limited to studying a trace in which peers are uniquely identifiable. Even though we have argued that our results should be generalizable to other trackers it is of importance to conduct further measurement studies to rule out the possibility of our results being influenced by chance.

A second example has been our limited knowledge of the field of experimental economics. Even though we have demonstrated experimental studies in this field that can model our indirect interaction mechanism we have not been able to identify games with settings that entirely matches our mechanism. In order to be able to predict user behavior, experiments need to be conducted in settings that better match our environment. While the option of implementing indirect interaction as the default behavior is a viable candidate users should be allowed to opt out of using IIM as a result of the privacy issues. Therefore it is important to predict user behavior when given such options. Better matching games within the framework of experimental economics could help us predict user behavior in such cases. Additionally, an interesting and important area of work involves the design of solutions to the privacy issue.

A third example of our limitation is the simple simulation tool that we have used to obtain our results. Future work is required to design and implement a more sophisticated simulation tool to study the properties of the mechanism under various other scenarios such as longer cycles of indirect interaction. Questions regarding the benefits of multiple indirect interactions also remain to be answered. Moreover, the question of whether the benefits of indirect interaction are cumulative still remains to be answered. In addition to the previous limitation recall that our simulation tool also lacks a method of modeling the properties of the underlying network (*i.e.* network delays). Future studies of IIM need to take the properties of the network into account. Ultimately a completely functional IIM enabled BitTorrent client and tracker are needed to study IIM in the real world.

Another interesting area of future work is to find the vulnerabilities of IIM. While we have considered a limited number of well known BitTorrent exploits and demonstrated that IIM is immune to most of these exploits we still lack an extensive study of methods by which IIM can be exploited.

Finally, we consider the most important area of future work to involve finding solutions to the scalability issue of IIM. Alternative tracker algorithms to find possible cycles could improve scalability as it is the main determinant in how well the mechanism scales. Randomized walks or approximation techniques could prove very effective and lead to solutions with higher scalability. Alternatively a detailed study of a distributed version of IIM is also important and necessary. A decentralized IIM could solve the scalability issue without the need of involving a central authority to find opportunities of indirect interaction.

Bibliography

[1] Tribler. Available at: `http://www.tribler.org/`.

[2] Bittorrent, the one third of all internet traffic myth. Available at `http://torrentfreak.com/bittorrent-the-one-third-of-all-internet-traffic-myth/`, September 2006. Accessed on September 22, 2010.

[3] Internet study 2008/2009. Available at `http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009`, 2008. Accessed on September 22, 2010.

[4] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. *First Monday*, 5(10), October 2000.

[5] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*, chapter 17-Complexity of Counting. Cambridge University Press, 2007.

[6] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. Analyzing and improving a bittorrent networks performance mechanisms. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, April 2006.

[7] Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. A game theoretic framework for incentives in p2p systems. *Peer-to-Peer Computing, IEEE International Conference on*, 0:48, 2003.

[8] Alice Cheng and Eric Friedman. Sybilproof reputation mechanisms. In *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 128–132, New York, NY, USA, 2005. ACM.

[9] Bram Cohen. Bittorrent protocol specification. http://www.bittorrent.org/.

[10] Bram Cohen. Incentives build robustness in bittorrent. Technical report, bittorrent.org, 2003.

[11] Rajdeep K. Dash, Nicholas R. Jennings, and David C. Parkes. Computational mechanism design: A call to arms. *Intelligent Systems*, vol. 18(no. 6):40–47, November 2003.

[12] John R. Douceur and Judith S. Donath. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.

[13] Kolja Eger and Ulrich Killata. Bandwidth trading in bittorrent-like p2p networks for content distribution. *Computer Communications*, Volume 31(Issue 2):Pages 201–211, February 2008.

[14] Ernst Fehr, Urs Fischbacher, and Simon Gächter. Strong reciprocity, human cooperation and the enforcement of social norms. *HUMAN NATURE*, 13:1–25, 2002.

[15] Ernst Fehr and Klaus M. Schmidt. A theory of fairness, competition, and cooperation. *Quarterly Journal of Economics*, Vol. 114(No. 3):817–868, August 1999.

[16] Joan Feigenbaum, Christos Papadimitriou, and Scott Shenker. Sharing the cost of multicast transmissions (preliminary version). In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 218–227, New York, NY, USA, 2000. ACM.

[17] Joan Feigenbaum and Scott Shenker. Distributed algorithmic mechanism design: recent results and future directions. In *DIALM '02: Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 1–13, New York, NY, USA, 2002. ACM.

[18] Michal Feldman and John Chuang. Overcoming free-riding behavior in peer-to-peer systems. *SIGecom Exch.*, 5(4):41–50, 2005.

[19] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust incentive techniques for peer-to-peer networks. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 102–111, New York, NY, USA, 2004. ACM.

[20] Michal Feldman, Christos Papadimitriou, John Chuang, and Ion Stoica. Free-riding and whitewashing in peer-to-peer systems. In *PINS '04: Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pages 228–236, New York, NY, USA, 2004. ACM.

[21] Urs Fischbacher, Simon Gächter, and Ernst Fehr. Are people conditionally cooperative? evidence from a public goods experiment. *Economics Letters*, 71(3):397–404, 2001.

[22] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang. Measurements, analysis, and modeling of bittorrent-like systems. In *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 4–4, Berkeley, CA, USA, 2005. USENIX Association.

[23] David Hales and Simon Patarin. How to cheat bittorrent and why nobody does. Technical report, Department of Computer Science University of Bologna, 2005.

[24] David Hales, Rameez Rahman, Boxun Zhang, Michel Meulpolder, and Johan Pouwelse. Bittorrent or bitcrunch: Evidence of a credit squeeze in bittorrent? 18th IEEE International Workshops on Enabling Technologies, 2009.

[25] Radu Jurca and Boi Faltings. Minimum payments that reward honest reputation feedback. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 190–199, New York, NY, USA, 2006. ACM.

[26] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM.

[27] John O. Ledyard. Public goods: A survey of experimental research. Technical Report 9405003, EconWPA, May 1994.

[28] Arnaud Legout, Nikitas Liogkas, Eddie Kohler, and Lixia Zhang. Clustering and sharing incentives in bittorrent systems. In *SIGMETRICS '07: Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 301–312, New York, NY, USA, 2007. ACM.

[29] Arnaud Legout, G. Urvoy-Keller, and P. Michiardi. Rarest first and choke algorithms are enough. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 203–216, New York, NY, USA, 2006. ACM.

[30] Dave Levin, Katrina LaCurts, Neil Spring, and Bobby Bhattacharjee. Bittorrent is an auction: analyzing and improving bittorrent's incentives. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 243–254, New York, NY, USA, 2008. ACM.

[31] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free riding in bittorrent is cheap. In *SIGCOMM '06*. HotNets, 2006.

[32] Sergio Marti and Hector Garcia-Molina. Taxonomy of trust: categorizing p2p reputation systems. *Comput. Netw.*, 50(4):472–484, 2006.

[33] 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*, 2010.

[34] Michel Meulpolder, Johan Pouwelse, Dick Epema, and Henk Sips. Bartercast: Fully distributed sharing-ratio enforcement in bittorrent. Technical report, Delft University of Technology;Parallel and Distributed Systems Report Series, 2008.

[35] Manfred Milinski, Dirk Semmann, and Hans-Jurgen Krambeck. Reputation helps solve the 'tragedy of the commons'. *Nature*, 415(6870):424–426, January 2002.

[36] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*, chapter 1; Basic Solution Concepts and COmputational Issues. Cambridge University Press, New York, NY, USA, 2007.

[37] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*, chapter 14; Distributed Algorithmic Mechanism Design. Cambridge University Press, New York, NY, USA, 2007.

[38] Martin A. Nowak and Karl Sigmund. Evolution of indirect reciprocity. *Nature*, 437(7063):1291–1298, October 2005.

[39] An Analysis of BitTorrents Two Kademlia-Based DHTs. Scott a. crosby and dan s. wallach. Technical Report TR07-04, Rice University, Department of Computer Science, Rice University, Houston, TX, USA, May 2007.

[40] Karthik Panchanathan and Robert Boyd. Indirect reciprocity can stabilize cooperation without the second-order free rider problem. *Nature*, 432(7016):499–502, 2004.

[41] David C. Parkes and Jeffrey Shneidman. Distributed implementations of vickrey-clarke-groves mechanisms. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 261–268, Washington, DC, USA, 2004. IEEE Computer Society.

[42] Michael Piatek, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. Do incentives build robustness in bittorrent. In *In NSDI07*, 2007. bittyrant.

[43] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. 2005.

[44] J.A. Pouwelse, J. Yang, M. Meulpolder, D.H.J. Epema, and H.J. Sips. Buddycast: An operational peer-to-peer epidemic protocol stack. Technical Report PDS-2008-005, Delft University of Technology Parallel and Distributed Systems, 2008.

[45] Dongyu Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 367–378, New York, NY, USA, 2004. ACM.

[46] Jelle Roozenburg. Secure decentralized swarm discovery in trible. Master's thesis, Delft University of Technology, 2006.

[47] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*, chapter 3; Introduction to Noncooperative Game Theory: Games in Normal Form, pages 47–88. Cambridge University Press, 2008.

[48] Karl Sigmund, Christoph Hauert, and Martin A. Nowak. Reward and punishment. *Proceedings of the National Academy of Sciences of the United States of America*, 98(19):10757–10762, 2001.

[49] Herbert A. Simon. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118, 1955.

[50] Michael Sirivianos, Jong Han Park, Rex Chen, and Xiaowei Yang. Free-riding in bittorrent networks with the large view exploit. In the 6th International workshop on peer-to-peer systems, 2007.

[51] Krit Wongrujira and Aruna Seneviratne. Monetary incentive with reputation for virtual market-place based p2p. In *CoNEXT '05: Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, pages 135–145, New York, NY, USA, 2005. ACM.

[52] Jiadi Yu, Minglu Li, and Jie Wu. Modeling analysis and improvement for free-riding on bittorrent-like file sharing systems. *Parallel Processing Workshops, International Conference on*, 0:53, 2007.

[53] Manaf Zghaibeh and Fotios C. Harmantzis. Revisiting free riding and the tit-for-tat in bittorrent: A measurement study. *Peer-to-Peer Networking and Applications*, Volume 1(Issue 2):162 – 173, July 2008.