

RICE UNIVERSITY

Designing Incentives for Peer-to-Peer Systems

by

Seth James Nielson

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Doctor of Philosophy

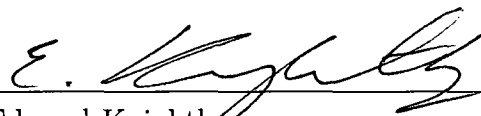
APPROVED, THESIS COMMITTEE:



Dan S. Wallach, Chair
Associate Professor of Computer Science
and Electrical and Computer Engineering



Devika Subramanian
Professor of Computer Science and
Electrical and Computer Engineering



Edward Knightly
Professor of Electrical and Computer
Engineering

Houston, Texas

October, 2009

UMI Number: 3421401

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 3421401

Copyright 2010 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

ABSTRACT

Designing Incentives for Peer-to-Peer Systems

by

Seth James Nielson

Peer-to-peer systems, networks of egalitarian nodes without a central authority, can achieve massive scalability and fault tolerance through the pooling together of individual resources. Unfortunately, most nodes represent self-interested, or *rational*, parties that will attempt to maximize their consumption of shared resources while minimizing their own contributions. This constitutes a type of attack that can destabilize the system.

The first contribution of this thesis is a proposed taxonomy for these rational attacks and the most common solutions used in contemporary designs to thwart them. One approach is to design the P2P system with incentives for cooperation, so that rational nodes voluntarily behave. We broadly classify these incentives as being either *genuine* or *artificial*, with the former describing incentives inherent in peer interactions, and the latter describing a secondary enforcement system. We observe that genuine incentives tend to be more robust to rational manipulations than artificial counterparts.

Based on this observation, we also propose two extensions to BitTorrent, a P2P file distribution protocol. While this system is popular, accounting for approximately one-third of current Internet traffic, it has known limitations. Our extensions use genuine incentives to address some of these problems.

The first extension improves seeding, an altruistic mode wherein nodes that have completed their download continue to provide upload service. We incentivize seeding by giving long-term identifiers to clients enabling seeding clients to be recognized and rewarded in subsequent downloads. Simulations demonstrate that our method is highly effective in protecting swarms from aggressive clients such as BitTyrant.

Finally, we introduce *The BitTorrent Anonymity Marketplace*, wherein each peer simultaneously joins multiple swarms to disguise their true download intentions. Peers then trade one torrent for another, making the cover traffic valuable as a means of obtaining the real target. Thus, when a neighbor receives a request from a peer for blocks of a torrent, it does not know if the peer is really downloading that torrent, or only using it in trade. Using simulation, we demonstrate that nodes cannot determine peer intent from observed interactions.

Acknowledgments

My thanks first and foremost to my Creator and for a chance to walk upon His footstool. It is ever a privilege and an honor to research the mysteries of His handiwork.

And, of course, my thanks to my wife Amy. I could not ask for a better partner. No matter how frustrated I was with my research, her joke about “it’s all just a for-loop” would put a big smile on my face. Thanks also goes to Alex, Drystan, Kael, and Saige who all suffered through this difficult journey. Without your support, I would not have made it.

This work was strongly helped by Johan Pouwelse who collected and shared traces of real BitTorrent swarms.

The component chapters were reviewed and critiqued by Matthew Green, Steven Bono, Gabriel Landau, Eugene Ng, Dan Sandler, and Scott Crosby. My thanks for the invaluable help in getting the thesis finished.

Finally, my thanks to my thesis committee members. To my adviser Dan Wallach who got me started on this path and saw it through to the end. Thanks also to Devika Subramanian for encouraging me in difficult decisions and helping me through tough academic spots. And to Edward Knightly for asking insightful questions that forced me to come up with better answers.

Contents

Abstract	ii
Acknowledgments	iii
List of Illustrations	vi
List of Tables	viii
1 Introduction	1
1.1 Background and Motivation	2
1.2 Contribution	5
1.3 Thesis Organization	6
2 A Taxonomy of Rational Attacks	8
2.1 Introduction	8
2.2 Economics Background	10
2.3 Model	11
2.3.1 Incentives Capabilities	11
2.3.2 Service Maturation	13
2.3.3 System Model	15
2.4 Taxonomy of Rational Attacks	16
2.4.1 Unrecorded Misuse of Resources	17
2.4.2 Unpunished Misuse of Resources	18
2.5 Solutions	19
2.5.1 Eliminate rationality as a concern	19
2.5.2 Design genuine incentives	20
2.5.3 Improving artificial incentives design	21
2.6 Conclusions	23
3 Long-Term Incentives in BitTorrent	24
3.1 Introduction	24
3.2 Background	26
3.2.1 The BitTorrent Protocol	26
3.2.2 BitTorrent Strategies	27
3.2.3 Ambient Altruism and BitTyrant	28
3.3 Incentives Design	30
3.4 Methodology	31
3.4.1 Simulator	31
3.4.2 Simulation Setup	33

3.4.3	Incentives Evaluation	35
3.5	Evaluation	39
3.5.1	Importance of Seeding	40
3.5.2	Rewarding Seeding	41
3.5.3	Bandwidth Reservation	42
3.5.4	Altruistic Population Size	44
3.5.5	Overlap	45
3.5.6	Seeding Rewards versus BitTyrant	46
3.5.7	BitTyrant Exploitation	47
3.6	Discussion and Future Work	50
3.7	Related Work	53
3.8	Conclusion	54
4	BitTorrent Anonymity Marketplace	57
4.1	Introduction	57
4.2	Background	59
4.2.1	BitTorrent	59
4.2.2	Incentives	60
4.3	Related Work	61
4.3.1	Tor	61
4.3.2	BitTorrent Specific Solutions	63
4.4	Design	64
4.5	Evaluation	67
4.5.1	Implementation	68
4.5.2	Development of the Valuation Function	69
4.5.3	Anonymity Results	72
4.5.4	Analysis	77
4.6	Discussion and Future Work	81
4.6.1	Stronger Anonymity and Ethical Issues	81
4.6.2	Informed Risk	84
4.6.3	Future Work	85
4.7	Conclusion	87
5	Conclusions	88
5.1	Genuine Incentives: Simplicity	89
5.2	Genuine Incentives: Impervious to Auditing Attacks	90
5.3	Genuine Incentives: Bounded Rationality	91
5.4	Genuine Incentives: Limitations	92
5.5	Genuine Incentives: Future Work and the Final Word	93
	Bibliography	95

Illustrations

2.1	Service Maturation Taxonomy	14
3.1	Simulated swarm membership over time based on a real-world trace from a flash-crowd swarm.	33
3.2	Cumulative distribution of efficiency (bandwidth utilization) over different populations in the same swarm.	37
3.3	Cumulative distribution of download time over different populations in the same swarm. (A different view of the same experiment shown in Figure 3.2.)	37
3.4	The median efficiency of the overall swarm under different compositions of clients. The worst performance is experienced when there is only one seed. When 70% of the clients seeding for 1-2 hours, the performance improves significantly. When 10% of the nodes seed for 1-2 days, the median efficiency approaches 100%.	40
3.5	Median efficiency when the altruistic population reserves 75% of the seeding bandwidth for other altruistic nodes.	42
3.6	Median efficiency as a function of the reserved bandwidth by the altruistic nodes.	43
3.7	Median efficiency as a function of the percentage of altruistic nodes in the swarm.	43
3.8	Median efficiency as a function of the percentage of overlap in the altruistic nodes.	43
3.9	Altruistic nodes versus tyrants under different amounts of reserved bandwidth.	48
3.10	Altruistic nodes versus tyrants with different ratios of altruistic nodes in the population.	48
3.11	Reward-seeding altruists, modified to trade tyrannically before they begin seeding, versus tyrant-leeches under different amounts of reserved bandwidth.	48
3.12	Median efficiency when altruistic nodes refuse to seed anything to tyrannical leech nodes.	49

4.1	The mean start rank of native interests plotted against popularity. The x -axis is the number of peers natively interested in the torrent, the y -axis is the starting rank. The error bars show the standard deviation. The wide standard deviations mean that native interests have a wide range of start rank.	74
4.2	The mean start rank of the various torrents plotted against the start rank for the same torrent for peers not natively interested. This graph shows that native interests do start sooner, but the mean lies within the standard deviation of non-native interest start times for most torrents.	75
4.3	Similar to Figure 4.1, this graph shows the mean ending ranks and the standard deviation. As with start times, end times vary sufficiently to make them poor predictors of interest.	75
4.4	The mean end ranks for native interest compare to mean end ranks for non-native interest. As before, there is a noticeable shift downwards, but, as before, the means for the native interests tend to lie within the standard deviations of the non-native interests.	76
4.5	Native and non-native traffic patterns super-imposed. While native traffic is above non-native traffic for the same node, the median for the native is within the standard deviation of the other.	78
4.6	This figure is similar to Figure 4.5 but limited to the viewpoint of single clients. In other words, the former figure is a global representation of download patterns, while this figure is representative of what a single peer observes.	79

Tables

3.1	Basic simulator performance as the number of simulated nodes (n) grows.	33
3.2	Comparison of median efficiency and median download time for the same experiment.	36
3.3	Median efficiency, averaged over twenty different experimental runs, differing only in the random seed.	36
3.4	Median efficiency, averaged over twenty experimental runs as above, with the leech nodes replaced by standard nodes.	36

Chapter 1

Introduction

The distinguishing feature of peer-to-peer (P2P) systems is the egalitarian responsibilities of the participants. Each node that joins the system is expected to contribute as well as consume shared resources. When the peers cooperate, the whole system can outperform traditional client-server systems in terms of scalability, resiliency, and fault-tolerance. Unfortunately, because most of the participants are *rational*, or motivated solely by their own interests, cooperation is not the default behavior and, without a centralized authority, it cannot be easily enforced. Selfishness drives nodes to consume more than their fair share of resources or refuse to give expected contributions. Behavior such as this, when contrary to system specifications, constitutes rational attacks, and these attacks can degrade performance or destabilize the system. To counter this threat, many P2P protocols are built around an incentives structure designed to make obedience the selfish choice because it is the only way to obtain maximum reward.

This thesis investigates principles of incentive design that increase the robustness of P2P systems to rational attacks. In this introduction, we first provide background and motivation for this effort as well as a sketch of our contributions to the research landscape. Chapter 2 then proposes a taxonomy of rational attacks and evaluates various defensive design principles including incentives. We then apply the insights of this evaluation to guide the development of two extensions to BitTorrent. These extensions for improved performance and improved anonymity are described in Chap-

ter 3 and Chapter 4 respectively. Chapter 5 subsequently presents our conclusions.

1.1 Background and Motivation

Traditional client-server computing is based on a central node or nodes that control the operation of the overall system. Client nodes may connect to, contribute to, and participate with these central servers but the servers are in control, and clients participate as *guests*. While clients may be allowed to communicate with each other, such inter-client communications are routed through the central authority where they are subject to review, filtering, and modification.

On the other hand, P2P systems [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] are built on an antithetical concept of node equality and direct peer-to-peer communication. Eliminating the central authority has tremendous benefits. The most immediate of these is that there is no single point of failure that can bring down the entire system. Scalability is also improved because, in general, the P2P system aggregates resources from the participants so that capacity is more or less proportional to load. On the other hand, servers must generally maintain a fixed resource profile that either struggles with peak use or is over-provisioned for average demand.

Despite these theoretical advantages, practical P2P systems completely deteriorate if and when participants refuse to cooperate. This so-called “free-riding” problem [11, 12, 13], is the general rule, unfortunately, because most nodes are rational, or utility-maximizing. Such nodes seek to maximize their own acquisition of shared resources while minimizing their own contributions. For example, consider a P2P system designed for content streaming [14]. To decrease the upload bandwidth burden on the original source, only a small number of nodes directly contact it. The content is then propagated from these nodes to additional peers. This system can only

scale if nodes obey specification and pass on content to downstream neighbors. The self-interested node, however, may simply decide not to expend upload bandwidth altruistically and refuse to retransmit the stream. Assuming that rational nodes are a large portion of the P2P population, it is likely that the system will not function well, if at all.

In most P2P systems, self-interested behavior at the expense of the system can be classified as a *rational manipulation* failure [15] or, from a different perspective, a *rational attack*. These attacks represent the primary and most fundamental challenge to P2P operations.

However, system designers have a powerful point of leverage in dealing with rational nodes. Unlike traditional attackers operating with the intent to cause harm through theft or vandalism, rational attackers are motivated by the benefits and services of the system. In other words, harming the system is not their primary motivation, but only an unfortunate side-effect. Therefore, if the only way to maximize utility is *through* cooperation, these nodes will voluntarily alter their behavior. Thus, a system designed with proper *incentives* [16, 17, 18, 19, 20] can achieve obedience despite the lack of a centralized enforcement authority. This approach is drawn from models of game theory and mechanism design [21]. We note, for completeness, that other forms of attack still matter and have been investigated [22], but are beyond the scope of this work.

Ideally, a P2P system should be perfectly *faithful* to the designer’s specification. In such a system, a self-interested, utility-maximizing node “will follow the default strategy because... there is no other strategy that yields a higher utility for this node” [23]. When this ideal cannot be achieved, the practical goal is simply to make the system sufficiently robust for continued viability. Determining what incentives

should be used in a system and how they should be employed represent a significant challenge to designers, and an intriguing academic problem.

One example of a P2P system that makes effective use of incentives is BitTorrent [1], a distributed download protocol that is popular in practice and research. The key incentive in the protocol is a tit-for-tat [24] exchange mechanism wherein peers trade different chunks of the file with each other. Peers reserve most of their upload bandwidth for rewarding their most generous neighbors. The remaining upload bandwidth is used to search for better partners by donating to other peers in the hope of being subsequently rewarded. While it has been shown that the BitTorrent protocol can be exploited [25, 26, 27, 28, 29], it has been sufficiently robust against rational attackers for practical purposes [23]. In fact, various measures of BitTorrent traffic have estimated that the BitTorrent protocol accounts for approximately one-third of all Internet traffic [30, 31, 32].

Nevertheless, the BitTorrent protocol can still be improved. For example, it permits nodes that have completed the download of the file to continue serving the swarm as a *seed*. BitTorrent requires at least one seed at all times or the other nodes cannot be guaranteed to complete the full download. In addition, seeds improve the overall performance of the swarm because they consume no resources, while still making contributions. Despite their importance, however, the default protocol offers no incentives for this mode of operation and many nodes contribute little if any seeding.

Another concern in the operation of BitTorrent is that of anonymity. BitTorrent itself was not designed to anonymize the participants in any way [1]. However, the design of the protocol is such that a node will expose its actions to potentially thousands [33] of peers, creating a visibility footprint that many users are uncomfortable with. One method for dealing with this problem is to pass BitTorrent traffic through

Tor [34]. *Tor* is a P2P anonymity network that routes traffic through a series of internal nodes to disguise the point of origin. Measurements [35] have shown that BitTorrent traffic represents a significant drain on *Tor* resources. Unfortunately, the incentives for contributing bandwidth resources to *Tor* are not clear and this results in low contributions and poor scalability. In the interest of improving these conditions, a few BitTorrent specific anonymity solutions [36, 37, 38] have been proposed, but none of them make effective use of incentives for cooperation.

1.2 Contribution

In this research, we answer the following questions:

1. **What are the general principles of incentives design that produce effective peer cooperation?**
2. **How can these principles be applied to BitTorrent’s seeding operation?**
3. **How can these principles be applied to the BitTorrent anonymity problem?**

The foundation of this research is our presentation of a taxonomy of rational attacks and corresponding principles of system design for ameliorating them. We answer our first key question by demonstrating that incentives can be broken into *genuine* and *artificial* categories. Genuine incentives are built into the P2P operations directly while artificial incentives are superimposed as a correction to the core protocol. Genuine incentives are generally more simple to understand and more robust to rational attacks.

Based on this principle of genuine incentives, our research then extends the BitTorrent protocol to incentivize the seeding operation. This extension modifies BitTorrent slightly to provide long term identifiers to participants. Nodes can then remember one another from one download to next. In this manner, nodes that seed in one swarm can be rewarded in later ones. Even though this extension requires a slight modification to the protocol, it can be implemented in a backwards compatible manner.

Finally, we also apply genuine incentives to the BitTorrent anonymity problem. Our extension creates a BitTorrent-like system that we term *The BitTorrent Anonymity Marketplace*. Inside this system, peers participate in many downloads at the same time to obscure their actual intention. In other words, the most powerful observer cannot reduce their actual intended download any further than 1 in k , where k is the number of simultaneous downloads. The key incentives design is that nodes compute a *value* for the cover traffic torrents they are trading. Nodes request torrents that they believe will help them download the item they really want. When a node makes a request from its peer, the peer cannot tell if the node really wants it or if it simply believes it can be used in trade. This incentives system ensures that any torrent in the Marketplace can be valuable to uninterested parties preventing starvation and strengthening anonymity.

1.3 Thesis Organization

This thesis is derived from three independent papers.

Chapter 2 is primarily derived from our paper, “A Taxonomy of Rational Attacks” by the author, Scott Crosby, and Dan S. Wallach. This chapter lays out how various types of attacks are connected to one another, the weak points of a system they exploit, and system design principles that improve them. Most importantly, it details

the nature of and the differences between genuine and artificial incentives.

Next, Chapter 3 is taken from, “Building Better Incentives for Increased Robustness in BitTorrent” by the author, Caleb Spare, and Dan S. Wallach. Contained in this part of the thesis is our proposed extension to BitTorrent as well as the introduction of a simulator we developed for evaluating the result. We also examine the behavior of BitTyrant, a strategic BitTorrent client, and its effect on our system. Part of our overall solution takes advantage of a BitTyrant bug we discovered.

The subsequent Chapter 4 is largely identical to, “The BitTorrent Anonymity Marketplace” by the author and Dan S. Wallach. Herein we present our design for the Marketplace, its implementation in simulation, and our results. This chapter also discusses significant ethical, legal, and moral issues that naturally arise in anonymity discussions.

Finally, in Chapter 5 we revisit our goals and our findings. This chapter ties together our results from the three component papers and discusses the significance. It also identifies opportunities for future work and several overarching questions raised by our research.

Chapter 2

A Taxonomy of Rational Attacks

For peer-to-peer services to be effective, participating nodes must cooperate, but in most scenarios a node represents a self-interested party and cooperation can neither be expected nor enforced. A reasonable assumption is that a large fraction of p2p nodes are *rational* and will attempt to maximize their consumption of system resources while minimizing the use of their own. If such behavior violates system policy then it constitutes an attack. In this chapter we identify and create a taxonomy for *rational attacks* and then identify corresponding solutions if they exist. The most effective solutions directly incentivize cooperative behavior, but when this is not feasible the common alternative is to incentivize evidence of cooperation instead.

2.1 Introduction

A significant challenge in peer-to-peer (p2p) computing is the problem of cooperation. Unlike client-server systems, a p2p network's effectiveness in meeting design goals is directly correlated to the cooperation of the member nodes. For example, a p2p system might be designed for content distribution. To decrease the upload bandwidth burden on the original content server, only a small number of nodes directly contact it. The content is then propagated from these nodes to additional peers. This system can only scale if nodes are willing to pass on content to downstream peers. Unfortunately, a self-interested node may realize that it can save expensive upload bandwidth if it chooses not to share. If a large number of nodes are self-interested and refuse to

contribute, the system may destabilize.

In most p2p systems, self-interested behavior at the expense of the system can be classified as a *rational manipulation* failure [15] or, from a different perspective, a *rational attack**. Successful p2p systems must be designed to be robust against this class of failure. Ideally, a p2p system should be perfectly *faithful* to the designer’s specification. In such a system, a self-interested, utility-maximizing node “will follow the default strategy because... there is no other strategy that yields a higher utility for this node” [23]. To achieve faithfulness, a system may employ various measures such as *problem partitioning*, *catch-and-punish*, and *incentives* [15]. Even when these techniques cannot make a system perfectly faithful, they may be enough to prevent destabilization.

An example of a viable p2p technology designed to be robust against rational manipulation failures is BitTorrent [1]. This technology first breaks large files into chunks that are downloaded individually and reassembled by the receiver. The receiving nodes contact one another and trade for chunks they do not yet possess. Each node employs an incremental exchange algorithm that leads it to upload chunks to cooperating nodes and not to share with selfish ones. These incentives encourage cooperative behavior in participating nodes [1]. While BitTorrent is not completely immune to rational manipulation, it is viable in practice [23].

In this chapter, we identify, analyze, and create a taxonomy of rational attacks in p2p systems. We then examine this taxonomy to identify corresponding solutions. In the next two sections, we first provide a short background on the economics principles applicable to p2p systems and then specify our system model. The following two

*Our definition for rational follows the narrow definition provided by Shneidman et al [15]. For the purposes of this chapter, rational participants are only interested in exploiting the resources and benefits of the system.

sections define our taxonomy of rational attacks and discuss solutions. The final section presents our conclusions.

2.2 Economics Background

Much of our analysis of p2p cooperation is based on economic models of game theory and mechanism design [21]. In this section, we briefly review some critical terms and concepts as they relate to p2p systems.

An economic *game* is a model of interaction between *players* in which the actions of any player influence the outcome of all other players. The *mechanism* in a game defines what legitimate actions the players can perform and the outcome of their behavior. These outcomes are assigned a numeric value called *utility*. Players that use an algorithm to determine behavior are said to follow a *strategy*

Players in the p2p world represent the nodes participating in the system. There are two types of nodes that do *not* strategize.

- *Altruistic* or *obedient* nodes cooperate with the system irrespective of any other considerations.
- *Faulty* nodes stop responding, drop messages, or act arbitrarily.

There are two types of nodes that do strategize.

- *Rational* nodes strategize to achieve maximal utility and their actions are based on their current knowledge and understanding of the p2p system. Rational nodes will not attempt to disrupt routing, censor data, or otherwise corrupt the system unless such behavior increases the node's access to shared resources. These nodes are also described as *self-interested*.

- *Irrational* nodes also strategize, but their strategies are either incomplete because they cannot understand the mechanism or they lie outside the economic mechanisms of the system. Denial of service or censorship attacks are examples of this second form of economically irrational behavior[†].

Mechanism design (MD) is the process of creating games where rational behavior by players leads to outcomes desired by the designer. Of course, such systems only affect the behavior of rational nodes. Mechanism design has no impact on faulty or irrational nodes and we exclude them from further discussion, though we recognize that any practical p2p system deployed “in the wild” must be resistant to their behavior. Of course, most p2p systems are robust against failure. The impact of irrational and malicious nodes is an open research problem that is discussed in Castro et al [22].

Distributed algorithmic mechanism design (DAMD) is a subclass of MD that is computationally tractable and operates without centralization. For this reason DAMD is well suited to systems like p2p networks [21]. DAMD assumes each node can independently reward the cooperation of other nodes or penalize their misbehavior but that each node has only limited information on the global state of the system.

2.3 Model

2.3.1 Incentives Capabilities

Incentives in p2p systems have some limitations. First, incentives are limited in the guarantees they can provide. While the use of incentives strengthens the p2p

[†]Our goal is to design systems which are immune to manipulation by nodes seeking increased shared resources. Our definition of rational only includes nodes whose utility function is independent of utility payout to other nodes. Strategies, such as censorship strategies, that obtain benefit by denying utility to other nodes are considered irrational.

system against rational attacks, by themselves they do not guarantee that the system is faithful. To be guaranteed faithful, a mechanism must be validated by a formal proof, the construction of which is not trivial.

The second limitation is that they must be DAMD compatible. DAMD is limited to creating mechanisms that are computationally tractable across distributed computing resources. Nodes are expected to reward cooperation and penalize misbehavior, but doing so is difficult when trusted global knowledge is unavailable.

With these two limitations in mind, we identify two types of incentives that may be used to create a faithful p2p system. The first type is *genuine incentives* and is characterized by directly incentivizing cooperation. A genuine incentive ties current behavior and future payoff together in some inseparable way. Genuine incentives are inherently robust against rational attacks and limit the strategies available to adversaries.

One example of genuine incentives is incremental exchanges as used in BitTorrent. Money could also be an effective genuine incentive but it would require very efficient micropayment schemes, where potentially every network packet transmission would require an associated payment. Unfortunately, the current generation of such systems (e.g., Millicent [39]) were never intended for such fine-grained commerce.

The second type of incentive is *artificial incentives*[†] which incentivize evidence of cooperation. Such incentives are weaker than their genuine counterparts because, to be rewarded, a node only has to *appear* to cooperate. Nevertheless, artificial incentives are generally easier to create and deploy and may be necessary under circumstances

[†]Roussopoulos et al. suggests that highly valuable shared resources have inherent incentives while less valuable ones require an extrinsic or artificial incentives for cooperation [20]. Our concept of genuine and artificial incentives is similar, but focuses only on the mechanism and not the value of the resources or social network in which the resources are exchanged.

where genuine incentives are not feasible.

Artificial incentives are often designed around an *auditing* process on top of which an enforcement mechanism is layered. In a decentralized system, auditing cannot be globally managed. Each node is aware of the system's policies, but is independently responsible for determining whether peers are in compliance. This can be done by requiring each node to publish assertions about its state which are audited by other nodes. An auditing policy of this type is consistent with DAMD; each node is capable of determining its behavior within the system. An auditing system, however, is subject to the vulnerabilities that we describe in Section 2.4.1.

2.3.2 Service Maturation

A p2p service provides some tangible benefit to participating nodes. New participants may obtain their payout spread over time, or they can obtain maximal benefit immediately in a lump sum. We have termed this service characteristic as *service maturation*. A service is mature when a node has obtained all of the benefit that the service can provide. Services that give out all possible benefit immediately have *instantaneous maturation* while services that distribute benefit over time have *progressive maturation*. Progressive maturation can be further classified as *bounded* or *unbounded* based on whether or not the service has a known, fixed termination of benefit pay-out. The relationship between the different classes of maturation is illustrated in Figure 2.1.

A content distribution service might have instantaneous or progressive maturation depending on policy. If a newly joined node can completely download its desired content before redistributing that content to peers, the service has instantaneous maturation. Conversely, BitTorrent has progressive maturation because it only al-

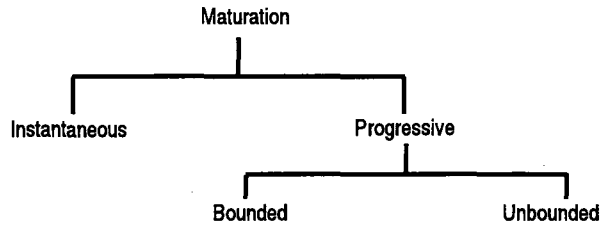


Figure 2.1 : Service Maturation Taxonomy

allows nodes to obtain the full content through repeated interaction with the system. Because BitTorrent’s pay-out of benefit ends when the file download is complete, its progressive maturation is bounded.

An example of a service with unbounded progressive maturation is a remote backup service. In such a system, the benefit payout is distributed over time without a fixed point of termination.

There is a correlation between instantaneous maturation to the Prisoner’s Dilemma (PD) and progressive maturation to the Iterated Prisoner’s Dilemma (IPD). In the single round PD, all of the utility that the game can pay out is disbursed in a single interaction. In IPD, the total utility is paid out to participants over some arbitrary number of interactions.

IPD also has an analog to the concept of bounded maturation. The game can be played with the players either aware or ignorant of the number of rounds that they will play. From the players’ perspective, the game is bounded only if they know the number of rounds. An IPD game degenerates into a PD game if the number of rounds are known.

Game theoretic analysis has proven that it is not rational to cooperate in single round PD but that it is rational to cooperate in IPD [24]. Services with instantaneous maturation are extremely susceptible to the attacks described in Section 2.4.2.

2.3.3 System Model

For convenience, we define a constrained environment suitable to explore rational attacks. The p2p model characterized in this section has many features that are common to most p2p networks. In Section 2.5 we break some of these assumptions as possible solutions to rational attacks.

Our model is described by the following assumptions and limitations.

Assumption: *Secure node ID's*. Douceur [40] observes that if identity within the p2p system is not centrally controlled, any participant can simultaneously assume a plethora of electronic personae. With many identities at its disposal, a participant can subvert the entire network by subverting the routing primitive. We assume that the node ID's in our model are made secure in one of three ways:

Trust - Node ID creation and distribution is done through a centralized and mutually trusted agent.

Expense - Node ID creation has some arbitrary cost attached. A participant can replace its node ID infrequently and with some difficulty.

Relevance - Node ID creation is unrestricted because having multiple ID's cannot aid the rational attacker.

Assumption: *There is no "trusted" software*. A p2p system cannot guarantee that their members are using conforming software. Trusted computing technologies allow a node to attest that it is running a conforming application [41, 42]. Enforcing a trusted software policy is not only technically challenging, but developing and deploying such a policy is undesirable to many

groups for ethical or practical reasons [43].

Assumption: *Nodes are computationally limited.* We assume that any given node may have the same resources as the typical desktop PC. Nodes may subvert their machine to behave in arbitrary ways. However nodes are assumed to be incapable of breaking cryptographic primitives or taking global control of the underlying network.

Due to the potential size of p2p systems and because nodes are in mutually untrusting domains, we apply the following limitations to our model.

Limitation: *Each node maintains minimal state.* A node can only have first-hand observations about a small fraction of the nodes in the system. Similarly a node can only maintain state about a small number of the nodes in the system.

Limitation: *No second-hand information.* Nodes can only trust what they directly observe because there is no inherent reason to trust an assertion by any node about a third party. An accusation can only be trusted if the evidence is independently believable regardless of trust in the accuser. Such proofs usually require the cooperation of the accused to create.

2.4 Taxonomy of Rational Attacks

The motive for the attacks we consider are unfairly increased access to p2p shared resources. We identify two general classes of attack:

1. Unrecorded Misuse of Resources
2. Unpunished Misuse of Resources

Attacks can be made by a single node, or by several nodes colluding together for an advantage.

2.4.1 Unrecorded Misuse of Resources

If an attacker can obtain resources without producing a record of the misuse, the attacker is safe from any sanctions. Attacks of this kind exploit “holes” in auditing policies (*policy attacks*), or actively disrupt the auditing mechanism (*auditing attack*).

Policy Attacks

A rational node may exploit an auditing policy. We identify two examples.

Excuses Any legitimate “excuse” for being unable to perform a service may be exploited. Such excuses may be needed to deal with edge conditions including crash recovery, network interruption, packet loss, etc. Consider a remote backup system like Samsara that requires every node to contribute as much space as it consumes [44]. If the system policy is overly generous to recovering nodes that recently crashed by not requiring them to prove they are maintaining their quota, a malicious node may exploit this by repeatedly claiming to have crashed.

Picking on the newbie Some systems require that new nodes “pay their dues” by requiring them to give resources to the system for some period of time before they can consume any shared resources [45, 46]. If this policy is not carefully designed, a veteran node could move from one newbie node to another, leeching resources without being required to give any resources back.

Auditing Attacks

Auditing attacks are designed to prevent the auditing system from identifying misbehavior. These attacks only apply to designs based around auditing using artificial

incentives. Here are a number of examples of this type of attack:

Fudged books Auditing relies on the accounting records being tamper-resistant and difficult to forge.

Manufactured evidence In this scenario, an attacker who is in a state of non-compliance manages to produce “proof” of compliance deceptively.

Accounting interruption (kill the auditor) A node being audited can attempt to interfere with the auditing node. This might be accomplished by a denial-of-service attack, a worm, a virus, etc.

Group deception, local honesty This attack is a type of manufactured evidence attack through collusion. Ngan, et al describes an accounting system where nodes publishing their debits and credits publicly in logs which are later audited by nodes’ peers [47]. Debts on one node must match credits on another node, making it more difficult for a node to cook its books. However, it is possible for single node in debt to become locally honest for an audit by pushing its debt to a co-conspirator. As a group, the conspiring nodes’ books are not balanced and they are in debt jointly. All colluding nodes reciprocate in sharing (or hiding) the debt.

2.4.2 Unpunished Misuse of Resources

An identified misbehaving node may attempt to avoid or mitigate punishment. Two such attacks are:

Elusion The attacker leaves the system permanently before they can be sanctioned by the p2p system. This attack generally exploits short-maturation and high-value resources. In such a scenario, the attacker obtains the resources and leaves

(e.g., join a content distribution service long enough to obtain an object and then disappear forever).

Reincarnation Reincarnation is repeated elusion. The attacker avoids punishment for misbehavior by assuming a new node ID thus releasing them from any penalties associated with its old reputation. We note that this attack is a limited form of the Sybil attack [40] where multiple ID's are acquired and discarded over time rather than all at once.

This class of attacks operates almost entirely against p2p services with instantaneous maturation.

2.5 Solutions

As stated previously, an ideal p2p system is perfectly faithful, but creating such a mechanism and proving its validity is difficult. In some cases a perfectly faithful design may be impossible, but a p2p system need not be perfectly faithful to be viable. In this section, we describe defenses against rational attacks by self-interested nodes in descending order of theoretical effectiveness.

2.5.1 Eliminate rationality as a concern

Under certain circumstances, forcing all nodes to be obedient may be practical and desirable. We identify three options for coercing obedience.

Out-of-band trust Obedience is enforced external to the p2p system. Such a scenario might be viable for a group of friends, or centrally administered machines within corporations, academic institutions, and government agencies.

Partial centralization It may be possible to introduce some aspect of centralization that induces nodes to be obedient. For instance a central authority can

be used to require secure node ID creation. BitTorrent uses a central authority to act as a rendezvous point where nodes can determine the addresses of their peers.

Trusted software - If a user is prevented from modifying their software, they must behave obediently. Many software applications are closed-source and difficult to modify. This may also be done with “trusted computing” technologies [48, 42].

2.5.2 Design genuine incentives

Genuine incentives are always preferred to artificial incentives. Because they are often difficult to implement in a DAMD context, it may be tempting for a designer to overlook them. Not only do genuine incentives eliminate many of the attacks described in Section 2.4.1, but they are also simpler than artificial incentives because they require no additional enforcement mechanisms.

For example, consider a back-up system with a storage policy similar to Samsara where each node must provide as much disk-space as it consumes in backups. One artificial incentives approach proposed by Fuqua, et al is to require that all nodes publish what data they are storing locally and to prove that they actually have that data in their possession on audit [47]. The auditing mechanism may be vulnerable to one or more of the auditing attacks described in Section 2.4.1.

A genuine incentive for the remote back-up service is to require that all of a node’s data that is stored on the network be tangled with the data it is supposed to be storing [45]. Nodes can then occasionally broadcast portions of the tangled data they are storing and ask for its owner to claim it or risk its deletion. Now the self-interested node must actually keep the data it claims to be storing or it cannot recognize claim-requests for its own data. However, to be useful, there must be a

policy that allows a node to reclaim its data after a crash even if it has lost all local-storage. This policy may expose the mechanism to the excuses attack described in Section 2.4.1. Despite this weakness, however, this mechanism is more robust and significantly simpler than the auditing alternative.

2.5.3 Improving artificial incentives design

Artificial incentives are a less desirable solution to rational attacks, but they may be the easiest to design into a service and are sometimes the only viable solution. Artificial incentives will generally entail having a well-defined auditing policy. A number of design decisions influence the effectiveness of these incentives.

Eliminating instantaneous maturation

A service which instantaneously matures is difficult to secure against rational attacks. Once a rational node has obtained the maximum benefit for a service, it has no incentive to continue participation. Thus, services that instantly mature are inherently vulnerable to elusion and reincarnation attacks. Also, because a node obtains its desired utility quickly, there is not much time for an auditing scheme to stop an attacker. Several techniques may help convert instantaneous to progressive maturation:

Centralized ID Creation If node ID's are centrally created and distributed, a node will be forced to maintain its identity in all of its future interactions with the p2p system. In this case if a node steals from the system and leaves, it will face punishment when it returns.

Security Deposit A node must contribute resources during a probationary period before it can benefit from the system's shared resources. Tangler is an example of system using this technique [45, 46].

Limited number of peers

Changing a node's ID incurs a cost. If an auditing system can detect and kick out a misbehaving node sufficiently fast, then the cost of changing identity outweighs the benefit. In most p2p systems, a node can only access the network through a limited number of neighbors. Once an attacker has freeloaded on its neighbors, they will refuse to interact with it and it will be effectively removed from the system. This solution has been used for multicast and storage accounting [18, 49, 50].

Reputation

With perfect global knowledge of every peer's behavior, a node would be incentivized to cooperate because any time it cheated, that information would be immediately available to all of its peers. Unfortunately, perfect global knowledge is only possible through an oracle which is not available in a DAMD context such as p2p networks.

Distributed systems may try to recreate the notion of a global, trusted oracle using gossip protocols, rating schemes, or some other form of peer endorsements. Mojo Nation had a global reputation system and EigenTrust describes how such systems might be built [51].

Protecting an auditing infrastructure

Because artificial incentives require building and protecting an auditing infrastructure, these mechanisms have additional complexity that may be prone to design and implementation errors. We suggest three practices for building effective auditing mechanisms:

Force the truth to be told Nodes can usually only believe what they observe for themselves. Secure history techniques [52], however, may be useful to generate

authenticated records of misbehavior that are trustable by remote hosts.

Double-entry bookkeeping A double-entry bookkeeping system as described earlier in Section 2.4.1.

Create a global clock When multiple nodes are being audited, they may be able to pass debts around from one node to the next, such that any particular node, while it is being audited, appears to have its books balanced. If several nodes can be simultaneously audited at provably the same time, this may defeat such attacks. Again, secure history techniques may provide an approximate solution to this problem.

2.6 Conclusions

In this chapter we explored a number of rational attacks. While we used a narrow definition of “rational”, we feel that this usage is justified by the unique nature of such attacks. From our analysis, we believe that designs that incorporate genuine incentives will generally be simpler and more robust than those with artificial incentives. Artificial incentives often require an auditing mechanism that is complicated and difficult to construct.

Unfortunately, given the difficulty of designing and implementing genuine incentives in a DAMD context such as p2p networks, artificial incentives will often be essential to incentivize cooperation for some parts of the system. When this is the case, avoiding instantaneous maturation eliminates unpunished misuse of resources attacks. A carefully designed policy and a robust auditing scheme are essential to mitigating unrecorded misuse of resources.

Chapter 3

Long-Term Incentives in BitTorrent

BitTorrent is a widely-deployed, peer-to-peer file transfer protocol engineered with a “tit for tat” mechanism that encourages cooperation. Unfortunately, there is little incentive for nodes to altruistically provide service to their peers after they finish downloading a file, and what altruism there is can be exploited by aggressive clients like BitTyrant. This altruism, called seeding, is always beneficial and sometimes essential to BitTorrent’s real-world performance. We propose a new long-term incentives mechanism in BitTorrent to encourage peers to seed and we evaluate its effectiveness via simulation. We show that when nodes running our algorithm reward one another for good behavior in previous swarms, they experience as much as a 50% improvement in download times over unrewarded nodes. Even when aggressive clients, such as BitTyrant, participate in the swarm, our rewarded nodes still outperform them, although by smaller margins.

3.1 Introduction

Peer-to-peer file transfer protocols provide scalable architectures for distributing large files. The core idea is to have peers participating in the download also contribute upload service back to the system, thus scaling the available bandwidth as more peers join. Even centralized services with large network connections can be overwhelmed by flash crowds, while p2p services can ostensibly continue to scale, even in such extreme scenarios.

In the practical world, however, scalability and stability in p2p systems are limited by the cooperation of the participants. These systems only have as much bandwidth as is collectively donated. Proper behavior cannot necessarily be enforced; participants are going to behave *rationally*, taking whatever steps maximize their own benefit without particularly caring about the well-being of other peers. Consequently, the default behavior of most participants is to consume and not contribute. This is often called the “free rider” problem.

BitTorrent [1] mitigates the free rider problem by rewarding uploads by granting faster downloads through a “tit for tat” (TFT) protocol, thus making cooperation a rational behavior. This design has been highly successful, enabling BitTorrent’s wide acceptance in the Internet community. While there is no consensus on the true amount of BitTorrent data in-flight today, it is clear that the number is large at somewhere between one-third and one-half of all Internet traffic [30, 31, 32, 53].

Despite the practical success of BitTorrent, numerous researchers have exposed weaknesses to the TFT incentives mechanism [25, 26, 27, 28]. One prominent weakness is the significant level of altruism that remains in the system despite the TFT mechanism. More specifically, many peers still contribute significant upload bandwidth without necessarily improving their download performance. Such contributions are produced by asymmetries in upload and download bandwidth as well as by altruistic BitTorrent behaviors like seeding and optimistic unchoking. (Section 3.2.3 discusses this “ambient altruism” in detail.)

These exploits are not simply theoretical. BitTyrant [25] takes advantage of the intrinsic altruism to achieve high download rates while reducing upload contributions. Most BitTorrent clients can be easily configured to rely exclusively on leeching, and some research suggests this is effective despite the TFT incentives [29, 27].

Our goal in this work is to reduce the altruism in BitTorrent seeding by adding incentives to the seeding component of the protocol. We present the design and evaluation of our seeding reward algorithm which requires a minor change to BitTorrent in the form of a long-term identifier for participating clients. Through simulation we demonstrate that rewarded peers get better performance than unrewarded peers. This differential creates an incentive for rational nodes to switch into the rewarded population. We further show that the rewarding mechanism improves node performance even when some portion of the swarm is composed of BitTyrant nodes.

In the remainder of the chapter, we first review the operations and altruism of BitTorrent in Section 4.2 as well as an overview of the BitTyrant variant. Sections 3.3 and 3.4 present our algorithm and the methodology we use to evaluate its performance. Our results are detailed in Section 4.5 and further analyzed in Section 4.6. We close with a discussion of related work in Section 4.3 and our conclusions in Section 4.7.

3.2 Background

BitTorrent [1] is a highly successful and popular peer-to-peer protocol which aims to enable efficient, rapid distribution of potentially large amounts of data to a group of clients. It is designed to utilize the available upload bandwidth of the clients to scale the capacity of the system to support many users and has built-in mechanisms to incentivize participation in this scheme.

3.2.1 The BitTorrent Protocol

A *torrent* is a file or a set of files users wish to download. The data is divided into equal-sized *pieces*, typically 256KB, which are further subdivided into small *blocks*. A central node called the *tracker* keeps track of the peers participating in the distri-

bution of a torrent. The tracker does not serve the actual content, but instead serves as a rendezvous point for peers to discover one another.

BitTorrent clients use a file of metadata, called a *torrent file*, to begin downloading content. This file, typically downloaded from a traditional web server, specifies the address for the tracker as well as information about the files to be downloaded, including names, sizes, and SHA-1 checksums for each piece.

The set of clients working on downloading a given torrent is referred to as a *swarm*. Clients notify the tracker as they join and leave the swarm, as well as every 30 minutes they are active within the swarm. To discover other clients, a client may query the tracker, which gives it a random subset of the active peers. (A variety of extensions exist which supplement the tracker, including a gossip protocol as well two DHT-based schemes.) Once it has a set of peers, a client establishes TCP connections to its peers, forming a *neighborhood* with whom it shares information about which pieces it has and has not completed downloading. A legitimate publisher might establish one or more official *seeds*, which provide round-robin, best-effort service to anyone who asks. These seeds are then supplemented by altruistic peers who seed after they finish their downloads.

3.2.2 BitTorrent Strategies

Popular BitTorrent clients employ a number of strategies to encourage fair participation in uploading and to deal with a variety of corner cases [1].

A client only uploads to a small number of peers in its neighborhood at any given time. This group of nodes is called the client's *active set*. The size of the active set is typically four, although both the reference implementation and BitTyrant [25] note that this number should scale with maximum upload bandwidth capacity. The

majority of the nodes in the active set are the nodes that have given the best service over a rolling 20 second average. The client saves one or two slots in the active set for the exploration of new neighbors. *Optimistic unchokes* pick a random peer every 30 seconds, allowing the client to search for better neighbors while also bootstrapping newly joined clients that have not yet downloaded anything to share.

BitTorrent clients share current status information with other clients to indicate which pieces are completely downloaded. Clients will bias their block requests to complete one piece before they begin downloading a different piece. To pick a piece to download, BitTorrent follows a *rarest first* policy, where a client picks pieces based on lowest availability within its neighborhood. The exception to this rule is for new clients, which need a complete piece before they can advertise any content for upload. In this case, they instead pick a random piece.

When a block has been requested, a client does not reissue the request until either the block is received or the request times out. This can be a problem when a user has received most of the pieces in a file and has just has a few outstanding requests to go. If the final peers are slow or unresponsive, the system might never finish. In this case, the client goes into *endgame mode* and sends redundant requests for any missing blocks to its peers; as they are received the client sends messages to the remaining peers to cancel unnecessary requests.

3.2.3 Ambient Altruism and BitTyrant

BitTorrent aims to reduce the free-rider problem, but it is not intended to eliminate altruism in the system. Instead, BitTorrent aims to ensure that a node will experience significantly improved performance if it participates in TFT trading, rather than leeching. Consequently, altruistic features remain in the protocol and pose two

separate, but related, problems. First, a client can reduce or eliminate its own altruistic participation, reducing the overall swarm performance. Second, if a client can recognize peers that are participating altruistically, it may be able to obtain sufficient service from these peers to find it unnecessary to deal with those that require cooperation.

Two significant sources of altruistic contributions are seeding and optimistic unchoking. Seeding is inherently altruistic under the current BitTorrent protocol. The altruism of optimistic unchoking is more complex. The optimistic unchoke operation is BitTorrent's method of searching the peer space for better TFT service. An unchoke that results in improved service because a better peer is found is clearly not altruistic, but unchokes are performed with random peers, rather than being biased away from known leeches. This means that BitTorrent's standard unchoking behavior can still provide a source of altruism, to the benefit of leeches.

Differences between peer bandwidth capacities also produce altruism. When a normal BitTorrent client unchokes a peer, it sends data as fast as the TCP stack will go, so peers with faster network connections will tend to give more out than they get in return when dealing with slower peers. Of course, two fast peers with content to trade will be more likely to establish TFT trading with one another than a fast peer and a slow peer.

BitTyrant is a strategic BitTorrent variant that exploits ambient altruism and reduces its own altruistic contributions [25]. BitTyrant was designed to download as fast as possible while contributing the minimum amount required to achieve it. To achieve this, BitTyrant abandons BitTorrent's policy of giving each member of the active set an equal share of its upload bandwidth. Instead, BitTyrant unchokes as many neighbors as possible but limits the speed of each upload stream to be only as

much as is necessary to obtain reciprocation.

This scheme does not work for other BitTyrant nodes, however, and two BitTyrant nodes must enter a special mode when dealing with each other. In Section 3.5.6, we will describe this special mode in detail and demonstrate how it can be used as part of a defense against BitTyrant’s behavior.

3.3 Incentives Design

Our incentives design for seeding in BitTorrent requires that the BitTorrent protocol support some form of long-term identifier. The basic concept for our algorithm is that BitTorrent clients recognize seeders from previous swarms and this is impossible without these IDs. Fortunately, the exchange of long-term identifiers can be built into the peer handshaking process in a backwards compatible fashion. Clients without a long-term ID are simply assumed to have no history. It is also worth noting that some clients [54] already support an optional long-term ID.

Our proposed design consists of an observation phase and a reward phase. The observation phase is in effect whenever the node is receiving seeding bytes, or bytes received from a neighboring peer without the expectation of TFT reciprocation. The detection of seeding bytes, in our basic implementation, is based on first-hand, verifiable information only. Obviously, it is possible that the neighbor is only pretending to seed, but from the observing node’s perspective, all bytes received without giving any bytes in return are seeded bytes.

The reward phase occurs when the node is in seeding mode. The goal is to schedule outbound seeding with higher priority given to peers who have seeded in the past. To do this, the algorithm first computes a score for each node; nodes who seeded get higher scores. These scores are used to initialize a scheduler, giving more slots to

nodes with higher scores. While virtually any scheduling algorithm would suffice, we chose to use lottery scheduling [55]. Each peer gets at least one ticket, but peers that seed get additional tickets in proportion to the logarithm of the number of bytes we have received from them in seeding.

Obviously, a node that chooses to be a good citizen and seed may not be rewarded at all in the future. For node A to be rewarded by node B , A must seed to B and then B must seed to A in some subsequent swarm. That means that both nodes must interact repeatedly over time. For any real benefit to the algorithm, a group of nodes must interact repeatedly.

We note that a Sybil attack [40] is possible against this protocol. For example, malicious nodes could create a large number of false identifiers, gaining additional shares of the bandwidth. We deal with this by reserving a percentage of a seeder's upstream bandwidth for other known seeders. Sybil attackers may well fight it out for the remaining unreserved bandwidth, but there is a larger pool of bandwidth available if they cooperate.

Another possible Sybil attack would be a *reincarnation attack* [56] where a client sheds an old identifier for a new identifier in every swarm to erase previously observed bad behavior. Such behavior would be unhelpful to the node, however, because a fresh identifier begins with no rewards at all. Rewards only come with observed good behavior.

3.4 Methodology

3.4.1 Simulator

We chose simulation as our primary method for analyzing incentives and altruism in BitTorrent. The advantages of a simulator over real world tests or the use of network

emulation lies primarily in the repeatability of the experiment and the time required to run the experiment. Our research requires comparison of algorithms against one another as well as experimentation with hundreds of combinations of parameters. Repeatability and fast time to completion were both incredibly helpful.

Several BitTorrent simulators exist but they did not fully meet our needs. One simulator from MSR [57] does not implement asynchronous communication nor does it capture some BitTorrent details, such as piece chunk transmission, that we deemed necessary. An ns-2 [58] BitTorrent simulator was also available, but it simulates TCP effects and other network level details that were too low level for our purposes. GPS [59] is a general purpose p2p simulator that includes a BitTorrent module and simulates at about the same level of granularity as our work. GPS is written in Java and our work appears to run faster.

To meet our objective, we have designed an optimized C++ simulator with a Python front end for simulation setup and execution. Our simulator allows swarms of thousands of clients, with several hundred running simultaneously, many times faster than real-time. To illustrate this, we ran a series of tests on an Athlon 2.4Ghz dual-processor server with 4GB of RAM and running with the Linux 2.6.9 kernel. These tests employed a simple swarm where a given number of clients arrive simultaneously and join the swarm. There is only a single seed for the swarm. We fix the file size at 100MB, the seed's upload capacity at 512Kbps, and each client's bandwidth at 56Kbps, symmetric for uploads and downloads. The results for various swarm sizes is shown in Table 3.1. These results show that the time required to simulate the swarm is proportional to the number of peers.

n	Sim Time (hours)	Real Time (hours)	Messages	Memory (MB)
10	5.86	0.004	233,950	20
100	4.77	0.07	1,381,715	60
1000	5.24	0.86	13,635,955	492

Table 3.1 : Basic simulator performance as the number of simulated nodes (n) grows.

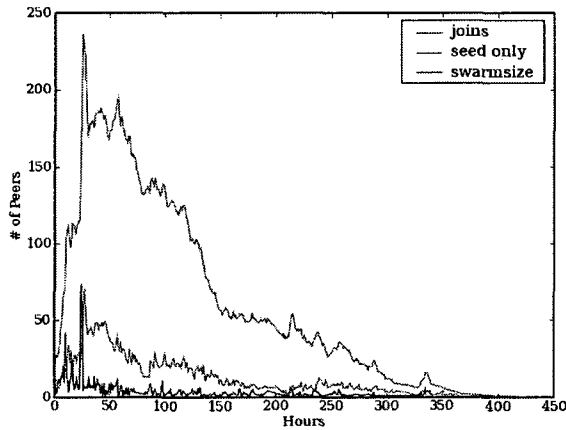


Figure 3.1 : Simulated swarm membership over time based on a real-world trace from a flash-crowd swarm.

3.4.2 Simulation Setup

All the evaluations in this chapter are based on a flash-crowd, 1GB file BitTorrent swarm. We used a total population of 2000 DSL clients with a range of download bandwidths from 128Kbps to 5Mbps. Each client's upload bandwidth is precisely half of its download bandwidth. To obtain reasonable churn, we make use of real-world BitTorrent traces taken in 2005 by Johan Pouwelse. These traces provide realistic join times for flash-crowd behavior in real swarms.

Each simulation is also configured with experiment-specific parameters. The significant parameters are:

Seeding Time The 2000 clients of the swarm are assigned one of three seeding population types. *Altruistic* clients will seed for 24 to 48 hours after their download is complete. *Standard* clients seed for one to two hours. *Leech* clients terminate their connection immediately after downloading the object. These values are based on why peers choose to seed; altruistic clients intentionally stay around to be helpful, standard clients will continue running until the user notices the download is done and kills the client, and leech clients leave as quickly as possible. Even though these numbers are guesses, we have validated that a swarm with 10% altruistic nodes and 70% standard nodes yields seed-to-swarm ratios similar to those observed in a prior measurement study. (Figure 3.1 in this chapter closely resembles Figure 5 in Pouwelse et al. [33].)

Seeding Algorithm Populations in the swarm can be assigned to use different seeding algorithms. The standard seeding algorithm simply seeds round-robin to all of the peers in a seed’s neighborhood. We also support an “incentives seeding” algorithm, as described in Section 3.3.

Incentives Seeding Parameters For peers using the incentives seeding algorithm, we can vary the bandwidth reservation for rewards as a percentage of the total bandwidth; all incentives seeding nodes will use the same reservation percentage in a given simulation run. Also, for nodes using our rewarding seeding algorithm, we invent a past history for each one, assigning them a number of bytes that they have seeded in the past. We similarly vary what portion of the population are aware of this history, allowing us to simulate everything from oracular knowledge of every node’s past behavior down to fragmentary knowledge that would be a more realistic approximation of prior, first-hand observations.

While oracular knowledge is unrealistic in practice, it allows us to place an up-

per bound on the benefits of seeding policies that use this knowledge. First hand information is more limited in scope but much more difficult to exploit [56]. In our research we are assuming that there are no disjoint cliques of overlapping peers. This would seem to adequately capture common classes of real-world behavior as we might expect from people who download related content, such as new episodes of TV shows released on a weekly basis.

Trading Algorithm We have implemented both the regular BitTorrent TFT and the BitTyrant trading algorithms in our simulator. Trading and seeding algorithms may be assigned independently; a peer can use the BitTyrant trading algorithm and our incentives seeding algorithm if desired.

3.4.3 Incentives Evaluation

Our goal is to create an incentive for participants in BitTorrent to seed. We will evaluate the effectiveness of our algorithm by demonstrating that rewarded populations perform better than unrewarded populations in our simulated swarms. By running the experiments under a variety of configuration parameters, we will characterize how these parameters affect the success of our incentives algorithm.

In evaluating the performance of a node, our basic measurement is the download efficiency, defined as the utilization of the peer’s download pipe over its lifetime in the swarm. Efficiency is a direct measure of the node’s happiness, and it is perfectly normalized. Any node, regardless of speed, cannot be happier than when it has 100% download utilization.

Computing the efficiency e is straightforward. Let k be the maximum download capacity of the node measured in bits per second (bps). Then let t_0 be the time the peer connected to the swarm and let t_d be the time that it finished the download,

Median Efficiency (%)			Median Download time (s)		
Altruistic	Standard	Leech	Altruistic	Standard	Leech
98.8	48.9	90.1	3304	7443	4402

Table 3.2 : Comparison of median efficiency and median download time for the same experiment.

Population	Average	Std. Dev	95% Confidence Interval
Altruistic	98.0%	1.8%	4.1%
Standard	57.9%	8.5%	15.3%
Leech	87.6%	4.8%	8.2%

Table 3.3 : Median efficiency, averaged over twenty different experimental runs, differing only in the random seed.

Population	Average	Std. Dev	95% Confidence Interval
Altruistic	97.9%	1.9%	4.6%
Standard	71.1%	7.9%	11.9%
“Leech”	71.2%	7.6%	12.9%

Table 3.4 : Median efficiency, averaged over twenty experimental runs as above, with the leech nodes replaced by standard nodes.

where both values are measured in seconds. Finally, let n be the number of bits in the download object. Then

$$e = \frac{n/(t_d - t_0)}{k}.$$

Of course, when simulating a large population of nodes with various configurations assigned at random, we would expect significant variation in individual nodes’ efficiency, even when they have the same configuration. Figure 3.2 shows cumulative distribution functions over nodes’ efficiency in a simulation with altruistic, standard, and leech nodes. A curve that stays closer to the bottom of the graph, as the altruistic data series does, represents more nodes operating closer to their peak efficiency. (This experiment shares the same configuration as used later in Figure 3.12.)

While we could potentially generate a figure like this for every possible simulation

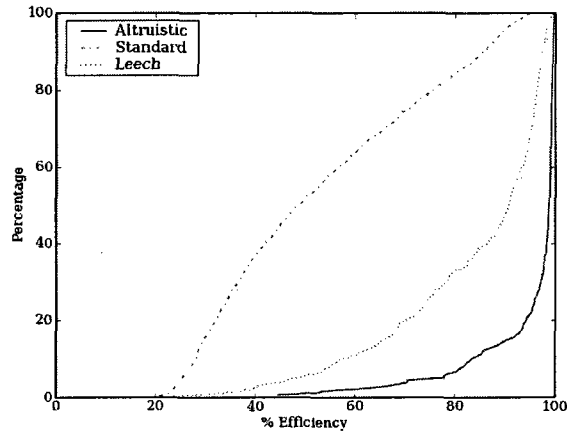


Figure 3.2 : Cumulative distribution of efficiency (bandwidth utilization) over different populations in the same swarm.

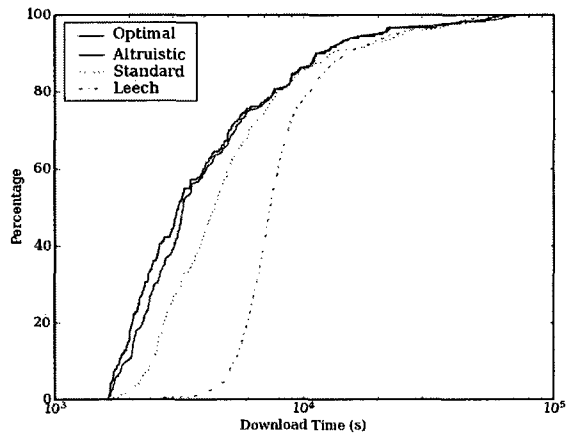


Figure 3.3 : Cumulative distribution of download time over different populations in the same swarm. (A different view of the same experiment shown in Figure 3.2.)

configuration, and every simulation run would generate a figure with the same general shape, this would obscure trends from one simulation to the next. Instead, we observe that the median value of each data series (i.e., the efficiency value for which the data series reaches 50% on the y -axis) represents an effective proxy for the overall behavior of the data. If the median values are close, then the curves will be close. If the median values are far apart, then the curves will be far apart.

For our experiments, then, any given set of experimental parameters (as described in Section 3.4.2) will yield three values: the median efficiency of each of the three populations (altruistic, standard, and leech), which we can then plot as we vary the simulation parameters.

An alternative to efficiency would be to consider the download times, without normalizing them for differences in each node’s available bandwidth. Figure 3.3 shows CDFs of download times for the same experimental setup as Figure 3.2. We added an “optimal” distribution, representing the best that the altruistic nodes could ever have performed if they had achieved 100% utilization of their download bandwidth. We could have added additional “optimal” lines for each population, but this would make reading the figure more complicated. Furthermore, median values are less meaningful because the underlying distribution of bandwidths would vary if the random assignment were done differently.

Of course, absolute download time and download efficiency are measuring the same underlying phenomenon; improving one metric would clearly improve the other. Table 3.2 shows the median values from each of these figures. The efficiency values elide unnecessary experimental details and concisely describe the relative performance of each population.

Lastly, we must convince ourselves that efficiency is a reliable metric from one

experimental run to the next. Since many of the parameters in our system are assigned randomly, we experimentally re-ran our experiment twenty times, each time with a different random seed. The results, shown in Table 3.3, show significant variation from one run to the next, but the variations among altruistic nodes are smaller than among standard nodes. For an additional experiment, we changed the leech nodes to be standard nodes. We would expect, then, that they would behave the same as standard nodes. Table 3.4 clearly validates this behavior.

From these measurements, it appears that standard nodes are more likely to be the victims of circumstance, while altruistic nodes and leech nodes are more stable in the face of random variation. As such, the reported performance of standard nodes should be considered to be noisier than the reported performance of altruistic or leech nodes. While we could precisely work out the minimum change between different populations that would represent a statistically significant difference, this is insufficient for our needs. Experimentally, we must show that our desired altruistic behavior doesn't just make a statistically significant improvement. We must show a large enough improvement to incentivize BitTorrent users to choose clients that follow our desired behavior.

(For the remainder of the chapter, we only run each experiment a single time for a given set of experimental parameters. Since each data point takes as long as a day to compute, we cannot afford to run every experiment twenty different times.)

3.5 Evaluation

In this section, we detail the findings of our research. We will first demonstrate why seeding is important for swarms of nodes with asymmetric bandwidth. We will then demonstrate how our algorithm improves performance for seeding nodes. The next

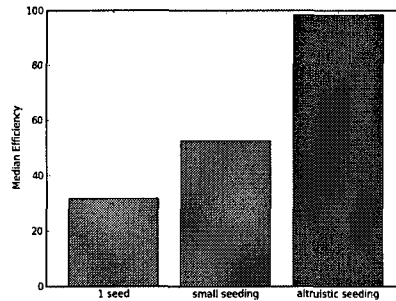


Figure 3.4 : The median efficiency of the overall swarm under different compositions of clients. The worst performance is experienced when there is only one seed. When 70% of the clients seeding for 1-2 hours, the performance improves significantly. When 10% of the nodes seed for 1-2 days, the median efficiency approaches 100%.

three subsections explore how bandwidth reservation, altruistic population size, and rewarding node overlap impact the effectiveness of our seeding algorithm. Finally, we analyze the performance of our algorithm in swarms that include BitTyrant nodes.

3.5.1 Importance of Seeding

Our first objective was to establish the importance of seeding to a BitTorrent swarm. We ran our simulation with three different population configurations. First, we ran the swarm with 1 initial seed and 100% of the swarm composed of our leech clients that do no seeding whatsoever. Next, we ran the swarm with 1 initial seed, 70% of the standard clients that do a small amount of seeding, and 30% of the leech clients. Finally, we ran a simulation with 10% altruistic nodes that seed significantly, 70% of the standard clients, and 20% of the leech clients. The results are shown in Figure 3.4.

There are two reasons why the swarm cannot obtain high efficiency without significant seeding contributions. First, the swarm is comprised of nodes with asymmetric bandwidth profiles. In our swarm, the upload is always half of the download capacity. Even with idealized operations, a swarm could hope for no more than 50%

efficiency from TFT trading alone. The second issue is that a BitTorrent swarm is not ideal. Various factors such as churn reduce the effectiveness of the protocol. Seeding provides enough additional capacity to overcome these deficiencies.

Clearly, seeding is essential for nodes in a swarm to maximize their download bandwidth; if we can design a mechanism that incentivizes more BitTorrent users to seed for longer periods, this should have a clear, positive impact on the system.

3.5.2 Rewarding Seeding

To evaluate our reward seeding algorithm, we first ran a baseline simulation. The setup for this simulation was 10% altruistic nodes, 70% of the standard clients, and 20% of the leech clients. All three populations were running the standard BitTorrent trading and seeding algorithms, thus we expected all three populations to experience similar performance. As expected, the results for all three populations was near 100% efficiency.

We then repeated this baseline experiment with all of the altruistic nodes configured to run our reward seeding algorithm, reserving 75% of their bandwidth for rewards to prior seeders. The other two populations continued to use normal seeding algorithms. In this version of our experiment, we assumed perfect overlap for this altruistic group. In other words, every altruistic node had been previously seeded by every other altruistic node, prior to the start of the experiment, and would thus allow the other altruistic nodes to share in the bandwidth reserved for rewards. The results of this simulation are shown in Figure 3.5. The altruistic population maintained nearly perfect efficiency, while the two unrewarded populations experienced a significant drop in performance.



Figure 3.5 : Median efficiency when the altruistic population reserves 75% of the seeding bandwidth for other altruistic nodes.

3.5.3 Bandwidth Reservation

As described before, our seeding algorithm can reserve bandwidth for the exclusive use of nodes being rewarded. To understand the necessity of these bandwidth reservations, we ran a simulation where we varied the percentage of reserved vs. unreserved seeding bandwidth. The results, shown in Figure 3.6, show the median efficiency of the altruistic, standard, and leech populations in simulations with different reserved bandwidth configurations. In all simulations, there are 10% altruistic, 70% standard, and 20% leech clients. The bandwidth reservation applies to altruistic nodes' seeding bandwidth. For the moment, we are assuming that altruistic nodes all have prior history and know which other nodes have seeded in the past.

One immediate observation is that our seeding algorithm, without any bandwidth reservation, does no better than normal seeding. This seems counter-intuitive because the rewarded nodes should still be getting more seeded bytes than their unrewarded peers. One might think that there would be some performance improvement for the altruistic nodes, even with 0% reserved bandwidth, but they are already getting nearly 100% efficiency.

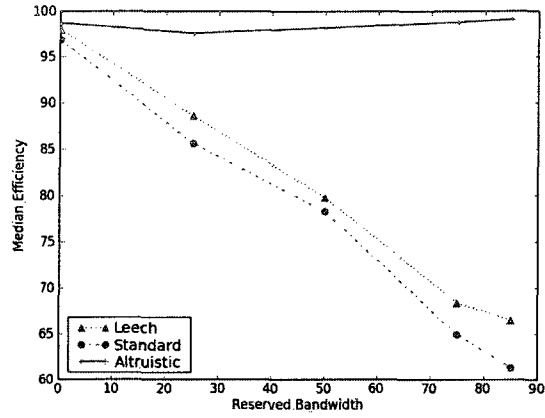


Figure 3.6 : Median efficiency as a function of the reserved bandwidth by the altruistic nodes.

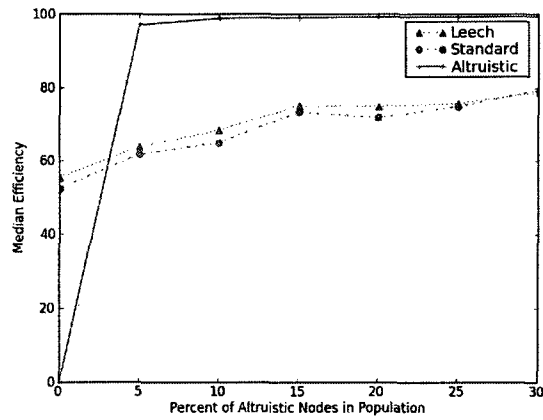


Figure 3.7 : Median efficiency as a function of the percentage of altruistic nodes in the swarm.

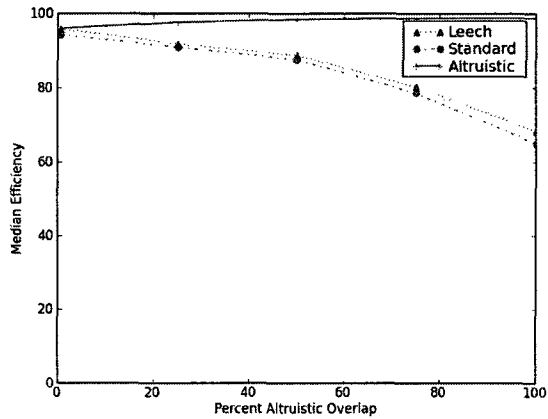


Figure 3.8 : Median efficiency as a function of the percentage of overlap in the altruistic nodes.

With bandwidth reservations, if there is insufficient demand from the “reward” population, then that portion of the seeding bandwidth will go unused. In short, our work suggests that the only way to create a performance differential between rewarded and non-rewarded nodes is to withhold service from unrewarded nodes.

There is an interesting trade-off, however. If the reservation is too high, then all of the bandwidth is effectively being spent on maintaining old relationships rather than establishing new ones. As nodes quit old swarms and join new ones on a regular basis, there is a clear incentive to have seeded to strangers in the past if there might be a payout in the future.

3.5.4 Altruistic Population Size

We cannot predict what percentage of nodes in a given swarm might be running our reward seeding algorithm. We would like to verify, regardless of the breakdown, that incremental growth in the reward seeding group will yield benefits both for those nodes as well as for everybody else. This leads to the question of how the system will respond as the population dynamics change. Figure 3.7 shows how efficiency changes as a function of the percentage of the altruistic and standard populations in the total swarm. The leech population is fixed at 20% and the rewarding nodes reserve 75% of their bandwidth.

This experiment demonstrates that the performance of the entire swarm improves as more nodes follow our altruistic scheme, even when reserving 75% of their bandwidth for reward seeding. That other 25% is enough to improve things for everybody else.

At some point, beyond the 30% altruism rate where we terminated our simulation, the standard nodes may have sufficient efficiency that they would be disincentivized

to change to the altruism strategy. By then, the altruism strategy would already be the dominant behavior in the swarm. Also, regardless of the rate of altruistic nodes, this experiment shows that altruism *always* wins, and sometimes wins big, even with relatively low populations of altruistic nodes.

3.5.5 Overlap

In this section, we explore the highly critical overlap parameter. Our algorithm assumes that nodes are rewarding based on first-hand information gleaned from prior interactions in prior swarms. In previous experiments, we have assumed that this knowledge of prior interactions, which we call *overlap*, is complete. Every node has prior, positive interactions with its altruistic peers and thus knows to include them in the reward population during future interactions. Such oracular knowledge is not realistic.

For simulation purposes, we wish to vary the degree to which altruistic nodes have had past interactions with other altruistic nodes and thus have the first-hand knowledge necessary to give reward seeding to their peers. To accomplish this, we partition the altruistic nodes into two sub-groups: rewarding and non-rewarding nodes. Rewarding nodes will reward all other altruistic nodes, including non-rewarders, while non-rewarding nodes will reward nobody. Non-rewarding nodes still have the same 75% bandwidth reservation, but they never use it. By varying the ratio of rewarding to non-rewarding nodes, we can roughly simulate the real-world effects that might be seen as the degree of overlap between altruistic nodes varies.

Figure 3.8 shows the efficiency for each population as a function of the percentage of altruistic nodes that are rewarders. We maintain a 10% altruistic, 70% standard, and 20% leech population. Reserved bandwidth remains fixed at 75%.

Our experiment demonstrates that overlap is clearly necessary to achieve the benefits of our reward seeding strategy. Once the overlap reaches 50% (i.e., about half of the seeding interactions between altruistic nodes are rewarded with higher bandwidth), the performance improvement for the altruistic strategy is undeniable. Whether such an overlap rate can be achieved in the real world is unclear. We discuss some strategies that might compensate for this in Section 4.6.

3.5.6 Seeding Rewards versus BitTyrant

In this section, we test the altruistic reward seeding algorithm against clients running the more aggressive BitTyrant trading algorithm. BitTyrant clients tend to see improved performance at the expense of other nodes in the system. (BitTyrant was introduced in Section 3.2.3.)

Our first experiment, shown in Figure 3.9, pits rewarding seeders against tyrannical leeches. This test repeats the bandwidth reservation experiment of Section 3.5.3 with the leeching population configured to use the BitTyrant trading algorithm. All other parameters remain the same.

Comparing these results against those of the earlier bandwidth reservation test, we note that BitTyrant-leeches performed as well as the rewarded altruists. At the same time the leeches degraded the performance of the standard nodes significantly. From this we conclude that the reward-seeding algorithm protects against, or at least ameliorates the exploitation of the BitTyrant protocol, but that it does not sufficiently penalize the leeching clients.

To evaluate how the size of the altruistic population impacts the performance of these populations, we repeated the experiment of Section 3.5.4, again with the rewarding altruistic seeders versus the tyrannical leeches. We hoped that increasing

numbers of altruists might be able to penalize the tyrannical leeches. Unfortunately, as shown in Figure 3.10, the tyrannical leeches still had no trouble achieving near perfect efficiency.

We considered the possibility that the leeching nodes would not do so well if the altruistic nodes were more stingy during the TFT trading phase. To test this, we reconfigured the bandwidth reservation test. In this experiment, the altruists use the BitTyrant TFT strategy rather than the default BitTorrent TFT strategy, but still perform the incentivized reward seeding. The leech population still practices tyrannical TFT trading and never seeds. The standard population uses standard algorithms for both seeding and TFT trading. All other simulation parameters remained the same. The results are shown in Figure 3.11.

Based on these experiments, a rational user might just as well run a tyrannical client as an altruistic client. They will receive the same download efficiency and they will minimize their upload bandwidth.

3.5.7 BitTyrant Exploitation

In the pursuit of finding a weakness in BitTyrant's seemingly anti-social behavior, we discovered a problem with BitTyrant's exchange mechanism (also noted by Carra et al. [60]). The original BitTyrant paper [25] says:

As such, BitTyrant continually reduces send rates for peers that reciprocate, attempting to find the minimum rate required. Rather than attempting to ramp up send rates between high capacity peers, BitTyrant tends to spread available capacity among many low capacity peers, potentially causing inefficiency due to TCP effects.

To work around this ... effect, BitTyrant advertises itself at connection

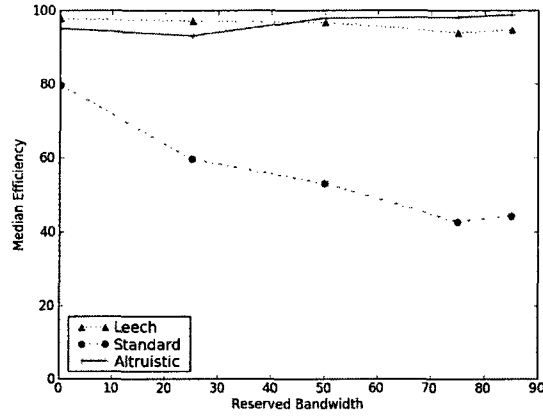


Figure 3.9 : Altruistic nodes versus tyrants under different amounts of reserved bandwidth.

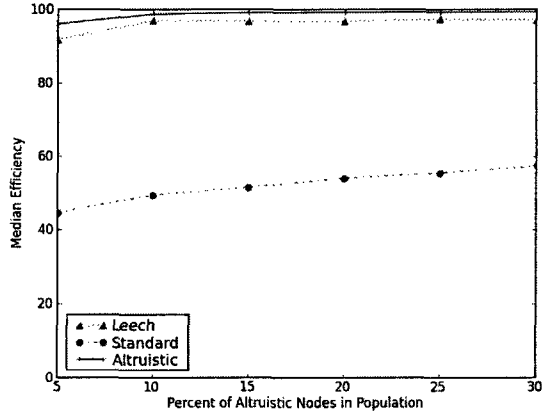


Figure 3.10 : Altruistic nodes versus tyrants with different ratios of altruistic nodes in the population.

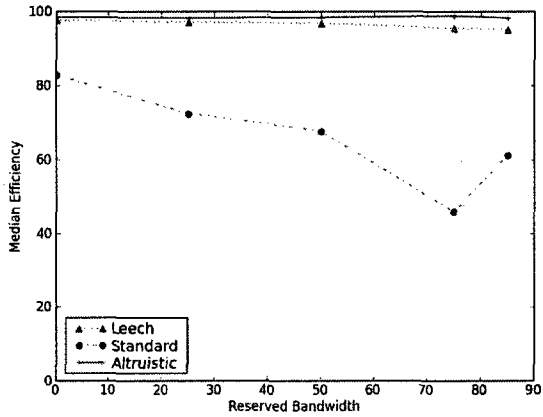


Figure 3.11 : Reward-seeding altruists, modified to trade tyrannically before they begin seeding, versus tyrant-leeches under different amounts of reserved bandwidth.

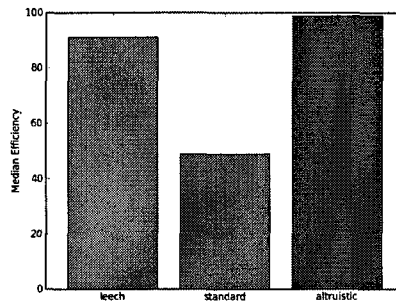


Figure 3.12 : Median efficiency when altruistic nodes refuse to seed anything to tyrannical leech nodes.

time using the Peer ID hash. Without protocol modification, BitTyrant peers recognize one another and switch to a block-based TFT strategy that ramps up send rates until capacity is reached.

The authors believe that their weakness is looking for too many low bandwidth flows, or that the many low bandwidth flows are inefficient because of TCP effects.

To evaluate this, we ran several simulations without the BitTyrant block-level TFT component (i.e., we disabled BitTyrant’s ability to detect that a peer is also running BitTyrant). BitTyrant nodes did very poorly when communicating with each other.

BitTyrant assumes it is receiving reciprocation when it receives an unchoke. This is a valid assumption for BitTorrent nodes, but it is not as clear of a signal from another BitTyrant node because it does not indicate how much they are willing to upload. So, if two BitTyrant nodes unchoke each other, they both assume they have an estimate for the minimum upload speed necessary to achieve reciprocation. They then both begin to drop their upload rates potentially down to zero in a quest to achieve lower estimates for the minimum upload speed.

BitTyrant solves this problem by self-identification, disabling the reciprocation-discovery mechanism because it doesn't really work between two tyrants. This identification features can be exploited by altruistic nodes to deny service to tyrants! A BitTyrant node cannot lie or obscure that it's a tyrant without incurring a penalty when trading with other tyrants.

We re-ran our baseline simulation with 10% altruistic, 70% standard, and 20% leech nodes. The altruistic nodes used the normal trade algorithm and our reward seeding algorithm. The leech nodes used the BitTyrant trade algorithm. Bandwidth was reserved at 75% and the altruistic nodes ignored tyrants during seeding, but interacted with them normally when still downloading the torrent. The results are shown in Figure 3.12.

By ignoring tyrants, the altruistic nodes achieve a small but significant performance improvement relative to the tyrants. There may well be other ways to exploit tyrants, such as refusing to interact with them at all. It is sufficient to say that BitTyrant is vulnerable to exploitation, itself, as a consequence of its necessary self-identification mechanism.

3.6 Discussion and Future Work

The development of this research gives rise to a number of important discussion points that we will address here. These points include issues relating to the practicality of our algorithm to real-life solutions as well as topics of future research.

Privacy / Anonymity is of significant concern for many BitTorrent users. Naturally, a long-term identifier would impact anonymity. However, the BitTorrent protocol was never engineered to provide anonymity to BitTorrent users. (They announce their presence to everybody in the swarm, based on their IP address, and adver-

tise what pieces they have available to trade!) From this perspective, a long-term identifier is not much worse than an IP address.

On the other hand, if a BitTorrent user chose to tunnel BitTorrent through an anonymization system like Tor, then the IP address would be obscured, while the long-term identifier would still be advertised. While a number of BitTorrent users do tunnel traffic through Tor, their performance will suffer greatly, as Tor was never intended to support the kind of massive, sustained traffic flows that BitTorrent can generate. Engineering an anonymity service specifically for BitTorrent would be an interesting opportunity for future research.

Bootstrapping and Overlap are the most critical concerns for further development of this incentives mechanism. The reward mechanisms in our research depend on the same nodes seeing one another, again and again. This may not occur much, in the general case, but it could well happen in particular subcommunities.

Existing Small Groups: A number of relatively small (compared to the entire world) communities exist for the purpose of BitTorrent distribution. The traces we described in Section 3.4 were collected from `filelist.org` over a three month period. This community requires a sign-in name which was associated with each download. We observed that 50% of all peers participated in at least two of the same swarms. These types of groups would be able to switch over to the seed-rewarding algorithm with very little difficulty and would likely have sufficient overlap.

Social Groups: Existing social communities, brought together by mutual interests on social networks, could be used to leverage a relatively small BitTorrent community suitable for the seed-rewarding algorithm.

Shared Interests: Even without explicit social groupings, one would reasonably expect that many people will follow similar patterns. For example, a variety of television shows are distributed via BitTorrent. Users who download the current show are likely to download subsequent shows. Similar affinities would be expected around other content that is updated on a regular basis, such as updated Linux distributions.

Transitive Trading and similar methods, may be able to ameliorate the need for extensive overlap. Transitive trading [49, 61] allows two clients that have never met to exchange “credits” through a mutual contact.

BitTyrant is an important development in BitTorrent because it improves the efficiency of certain core concepts. For example, the optimistic unchoke in standard BitTorrent trading is a *search* method for finding better peers, but it simply searches randomly. However, as we discussed in Section 3.5.7, BitTyrant clients must identify whether they are speaking to other tyrants and change strategies. Otherwise, the default BitTyrant TFT strategy will have both clients dropping their bandwidth all the way to zero.

This BitTyrant flaw creates interesting opportunities. Since BitTyrant clients must identify themselves as such, they can be trivially ignored by other clients who, perhaps, do not wish to support their tyrannical behavior. However, there are many other options. BitTyrant clients (or, really, any BitTorrent client) could publish categorical statements about their unchoking policies. For example a node might declare: “If you give me at least X bytes per second, then I’ll unchoke you and give you X in return, up to Y bytes per second max.” Of course, a tyrant could lie about such policies, but it creates an interesting opportunity for future research, both in

terms of simulation studies and in terms of economic modeling.

Carra et al. [60] also examined the strengths of BitTyrant-style behavior versus simply expanding the number of simultaneous connections in traditional BitTorrent clients by simulation. However, their simulation models ignored churn and other real-world conditions leading us to believe that their results are unreliable.

3.7 Related Work

The BitTorrent protocol and associated algorithms were introduced by Cohen in 2003 [1] with a reference client implementation. A fluid model for the system was given by Qiu et al. [62], who used it to show that in certain cases a Nash equilibrium can exist in systems where peers choose upload rates to match their download rates. Studies performed on emulated swarms by Legout et al. [28] validated the effectiveness of the BitTorrent unchoking algorithm. Legout et al. [63] also concluded from real-world tests that the rarest-first algorithm is very important to system performance, and argued that the default unchoking algorithm provides adequate robustness from free-riders.

A fluid-model simulator was used by Bharambe et al. [57] to represent a BitTorrent system in a more abstract manner than our own. They confirmed the utility of the rarest-first policy for piece selection. They also investigated unfairness with respect to volume uploaded and argued that the rate-based TFT strategy fails to prevent such unfairness, especially in systems with a great disparity of bandwidth among peers. They proposed a new block-level, volume-based TFT trading algorithm, although subsequent researchers challenged its effectiveness [63].

De Vogelee et al. [64], made an event-based simulator for BitTorrent based on the algorithms in the reference implementation and used it to model a variety of typ-

ical swarm scenarios, verifying the performance characteristics against the expected behavior of a standard BitTorrent client.

A simulation-based study by Eger et al. [58] compared flow-level and packet-level simulations for BitTorrent-like systems and found that, while flow-level simulations are useful for demonstrating the theoretic performance of the de facto BitTorrent scheme, the delay of TCP packets and other cross-layer effects have a significant impact on BitTorrent performance, and these effects require a more granular simulation to be adequately captured.

Much research has been performed concerning the robustness of BitTorrent's tit-for-tat trading mechanism against selfish behaviors. BitTorrent was modeled as a form of the Iterated Prisoner's Dilemma problem by Jun et al. [65], who suggested that the current peer-selection algorithm is susceptible to free-riders; they proposed a different TFT strategy. Tian et al. [26] used mathematical models as well as simulation-based and real-world experiments to argue for a modified TFT algorithm.

Sirivianos et al. [27] emulated a strictly free-riding client which contacts the tracker often to gain a large neighborhood from which to free-ride; they concluded that this attack was feasible in practice. Liogkas et al. [28] use PlanetLab to demonstrate three different exploits: downloading from seeds, downloading from the fastest peers, and advertising fake pieces.

3.8 Conclusion

At present, BitTorrent's seeding mechanism is entirely altruistic; nodes have no rational reason to offer seeding service to their peers, yet the additional bandwidth provided by seeding is essential to the efficient operation of BitTorrent. Anything that can encourage seeding would have an immediate knock-on benefit for BitTorrent

users.

In this work, we have proposed a method for rewarding seeding in BitTorrent by means of long-term identification. Nodes remember peers that seeded to them in the past and reciprocate by seeding to them in later swarms.

To evaluate our algorithm and its parameter space, we developed and employed a flow-level simulator. The algorithm was tested on realistic file-sizes and trace-driven churn to improve its accuracy. We found that our algorithm improved the download efficiency of the BitTorrent nodes from 70% to 95% or better. This improvement represents the upper bound of our algorithm's performance and was based on oracular knowledge that would not be available in real scenarios. We tested more realistic settings and found that our algorithm could still increase the download efficiency by ten percentage points.

Finally, we evaluated our seed-rewarding algorithm in swarms that had some portion of the population running BitTyrant, a variant on BitTorrent that is aggressive about getting fast downloads with minimal investments of upload bandwidth. We found that our algorithm could protect nodes from being exploited by BitTyrant, but could not sufficiently penalize tyrannical behavior to discourage users from choosing to run BitTyrant. However, leveraging a weakness in BitTyrant, where BitTyrant nodes must identify themselves as such, we can ignore tyrants during seeding and reduce their performance.

So long as BitTorrent peers have sufficient overlap in successive swarms, allowing them to build individual long-term histories of who has seeded in the past, we conclude that BitTorrent peers using our incentivized reward seeding algorithm will enjoy better performance for themselves and also improve performance for their peers, whether running our algorithm or not. By adding in our mechanism, for which peers

have a genuine incentive to follow, we can build better robustness in BitTorrent.

Chapter 4

BitTorrent Anonymity Marketplace

The very nature of operations in peer-to-peer systems such as BitTorrent exposes information about participants to their peers. Nodes desiring anonymity, therefore, often chose to route their peer-to-peer traffic through anonymity relays, such as Tor. Unfortunately, these relays have little incentive for contribution and struggle to scale with the high loads that P2P traffic foists upon them. We propose a novel modification for BitTorrent that we call the *BitTorrent Anonymity Marketplace*. Peers in our system trade in k swarms obscuring the actual intent of the participants. But because peers can cross-trade torrents, the $k - 1$ cover traffic can actually serve a useful purpose. This creates a system wherein a neighbor cannot determine if a node actually wants a given torrent, or if it is only using it as leverage to get the one it really wants. In this chapter, we present our design, explore its operation in simulation, and analyze its effectiveness. We demonstrate that the upload and download characteristics of cover traffic and desired torrents are statistically difficult to distinguish.

4.1 Introduction

Peer-to-peer file transfer protocols, such as the very popular BitTorrent [1] protocol, provide massively scalable architectures for distributing large files. Unfortunately, privacy is a direct casualty of the peer cooperation that drives them. For traditional client-server architectures, the client need only trust the server not to reveal to additional parties the details of the transaction. While some information is revealed just

from observing that the client and server communicated with each other, the specifics are confidential. With appropriate cryptographic and protocol mechanisms, the client can have strong assurances of privacy in the transaction so long as the server remains trusted.

On the other hand, in peer-to-peer cooperation, an individual, by necessity, reveals details of the transaction to many parties, each of which must be trusted if privacy is to be maintained. This problem is exacerbated by the nature of peers in such systems. In the client-server model a user can limit interactions to well-known, vetted servers, but in contemporary p2p systems peers could be controlled by an incompetent or malicious individual or organization.

A number of solutions to the peer-to-peer anonymity problem have been proposed. The most common solution in practice is to route traffic through anonymity relays such as Tor [34]. Unfortunately, Tor has, by default, no incentives for cooperation and struggles to scale with P2P workloads. Our goal at the onset of this research was to develop an anonymity mechanism for BitTorrent that incentivizes participation and induces scalability. Such a mechanism would provide better performance than BitTorrent-over-Tor while still providing sufficient anonymity guarantees. Furthermore, it would draw BitTorrent users away from the Tor network and all parties would be better off.

We have created the *BitTorrent Anonymity Marketplace* as novel solution to this problem. This system provides genuine incentives for nodes to participate in cross trading of multiple swarms obscuring the actual intent of the driving nodes creating what we refer to as *k-traffic-anonymity*. We demonstrate in simulation the effectiveness of this obfuscation and show that it has nearly optimal performance tradeoffs. Our result is distinguished from other BitTorrent specific anonymity solutions either

because participation is incentivized, or because the attack model we address is more powerful.

This chapter is organized as follows. We first review some of the operations of BitTorrent and some of the principles of incentives in Section 4.2. In Section 4.3 we review the current solution space to the p2p anonymity problem. Then we introduce our own objectives and design in Section 4.4. We evaluate our results in Section 4.5. Finally, we close with a discussion of our research in Section 4.6 and our conclusions in Section 4.7.

4.2 Background

4.2.1 BitTorrent

BitTorrent [1] is a highly successful and popular peer-to-peer protocol that enables efficient, rapid distribution of potentially large amounts of data to a group of clients. It utilizes the available upload bandwidth of the participants to scale to support many users. Most important, it has built-in incentives mechanisms that reward correct participation.

To make an item available for BitTorrent downloading, a publisher makes available a *tracker* and at least one *seed*. The tracker follows the nodes participating in the swarm, helping nodes locate their peers. Seed provide round-robin, best-effort service to all connecting peers.

To download the object, a group of nodes, collectively called the *swarm* join the system by contacting the tracker, indicating their intent to participate. The tracker informs joining nodes of random subsets of their peers. The nodes then establish direct connections with these subsets forming their local *neighborhoods*. Thus joined, the nodes download the object in equal sized chunks of the file called *pieces*. Nodes

share information with their neighborhood about the pieces they have available and update them as new pieces are acquired.

Nodes, however, limit the number of peers in their neighborhood that can download from them at any given time. They evaluate their peers based on how much each has recently uploaded. The node then provides download service to the top three or four contributors. Each node also provides service to one or two random nodes as a method of searching the neighborhood for better partners. Thus, peers have an incentive to contribute to their neighbors in order to receive reciprocal contributions from their neighbors in turn. When a node decides to service a peer, it is said to *unchoke* the peer. Conversely, when it will no longer serve the peer, it is said to *choke* it. Once a peer is unchoked, it can send *Request* messages asking for data. If the unchoking node refuses, the peer considers itself snubbed and will not do business with that node for some time. Nodes update their peers with *Have* messages when a new piece is received so that the neighborhood keeps abreast of what a node can and cannot trade.

While a significant corpus of research has demonstrated that BitTorrent can be exploited [25, 26, 27, 28], BitTorrent continues to work well in practice. The incentives in BitTorrent are sufficient, at present, for keeping the system stable. Indeed, while there is no consensus on the true amount of BitTorrent data in-flight today, it is clear that the number is large at somewhere between one-third and one-half of all Internet traffic [30, 31, 32, 53].

4.2.2 Incentives

Peer-to-peer systems' greatest strength is their lack of centralization. At the same time, this lack of centralization makes enforcement of peer behavior difficult. In

general, the system designers intend for peers to behave in a certain way, but peers may choose to behave differently. Most nodes are assumed to be *rational*, or self-interested, and want to maximize their benefit from the system while simultaneously minimizing their own contributions. *Faithfulness* is the measure of a node's obedience to designer specification. By definition, rational nodes are only faithful if it is in their own interest, and, therefore, faithfulness can only be achieved by designing systems with proper incentives [15, 23].

In previous work, we identified two general classes of incentives in peer-to-peer systems: artificial and genuine [56]. Genuine incentives are characterized by being an intrinsic property of the p2p protocol, whereas artificial incentives are a superimposition of reward and punishment on top of an unincnetivized system. The intrinsic nature of genuine incentives makes them more robust to rational manipulations and are, therefore, preferred.

4.3 Related Work

A number of solutions to the peer-to-peer anonymity problem exist or have been proposed. We briefly outline some of these approaches here.

4.3.1 Tor

Tor [34] is distributed network of relays operated by volunteers that allows clients to route network traffic through them to disguise the true origin. If used properly, the client's identity and physical location are kept hidden from other entities. Per-relay encryption also provides anonymity against wire-traces and packet sniffing. Each relay is allowed to define its own policy about what it will and will not do for the network. *Entry routers*, as the name implies, accept traffic from outside the Tor network. Conversely, *exit routers* allow traffic out to the true destination. *Middle*

routers only relay traffic within Tor itself.

A node that desires anonymity computes an *onion route* through the Tor network. It encrypts its packet with a layer of encryption for each router in the network. Each intermediate node peels off a layer of encryption, and forwards the traffic to the next hop. Each node only knows the preceding and subsequent steps in the route. The nodes cannot be sure if the packet they are receiving is from the original sender, or simply a relay in the route.

Measurements taken in [35] indicate that 40% of the traffic from a sample Tor exit node was used for BitTorrent indicating how popular Tor is for providing BitTorrent anonymity.

Despite Tor's usefulness, it does struggle with a significant problem. It has trouble encouraging participants to contribute new computers to serve in the Tor network, impacting Tor's ability to scale with the traffic it receives. Additional nodes also strengthen anonymity. However, the value of serving as a relay to a user is unclear; it has no impact on the quality of service that they observe from the Tor network. Consequently, most users choose not to contribute.

Another important observation is that any negative legal or social response resulting from the originator's connection will be borne by the exit node. Consequently, many nodes have a strict disincentive to not serve as an exit node.

Artificial Incentives for Tor Recently, researchers have proposed extending Tor with incentives for better participation. One proposal [66] is to create a central authority that tracks each node's contributions and publicizes their good behavior so that other nodes can reward them. Alternatively, other research proposes micropayments, where Tor users may buy a higher quality of service [67].

4.3.2 BitTorrent Specific Solutions

In addition to the Tor general anonymity network, anonymity mechanisms have been proposed that are specific to BitTorrent.

BitBlender [36] extends BitTorrent to route traffic through peers in an anonymity directory. In a fashion similar to Tor, members of the swarm can forward requests through other peers providing a form of anonymity it calls “ k -anonymity.” They define this as “users are indistinguishable from a set of k users.” Unfortunately, as with Tor, BitBlender provides no incentive for nodes to offer relay services. Please note that k -anonymity in their system is not the same as k -traffic anonymity in this chapter.

OneSwarm [37] attempts to solve the BitTorrent anonymity problem more generally. Nodes have extensive control over what information about themselves they will share and with whom. In particular, OneSwarm would be used with social networking so that information is only shared with “friends.” OneSwarm does not solve the problem of maintaining anonymity in large groups of untrusted peers.

SwarmScreen [38], in a fashion similar to our work, proposes anonymity through the use of cover traffic. In particular, they assert that nodes achieve plausible deniability “by simply adding a small percent (between 25 and 50%) of additional random connections that are statistically indistinguishable from natural ones.” SwarmScreen’s attack model has an observer classify nodes based on the behavior of the community they participate in. Its stated goal is the disrupting of these “guilt-by-association” attacks, or in other words, obscuring the *community* that a node is participating with at any given point in time. We will make further comparisons to SwarmScreen as we

outline our own solution. Our work is only superficially similar.

4.4 Design

Our objectives for this work break down into three categories: anonymity, performance, and incentives. As we detail our objectives, we will compare and contrast our solution with SwarmScreen to illustrate the differences in approach and philosophy.

Our primary goal is an obfuscation of participant behavior that we call *k-traffic-anonymity*. Nodes in our system must have an indistinguishability of intent as they are observed by their peers. In other words, a node's peers can see that they are downloading k items but cannot distinguish which one of them the node picked intentionally. The intentionally picked torrent is called the *native interest*.

Our primary threat: observers wish to ascertain a target node's native interest. We call the attacker an *inquisitor* and define three different classes of attacks. *Fully passive* inquisitors do not contact any other peers. Instead, these nodes exclusively scan the tracker's data on where nodes are participating. *Decoy passive* inquisitors do contact peers and can appearance to participate. They may lie and announce piece reception, make requests for pieces from their peers, and in any other way appear to be normal nodes, but they will not actually accept downloads or make uploads. Finally, *Active* inquisitors can participate and behave like any other node in the system.

Within our anonymity constraints, we want good performance. We will measure performance in terms of the number of additional download bytes required to achieve a given level of anonymity. In an idealized world where all torrents are the same size, optimal performance for *k-traffic-anonymity* is k times the number of bytes in a torrent. In other words, the node downloads exactly k torrents and nothing more. Our objective is nearly optimal performance; we are not interested in designs, for

example, that require $2k$ or more download cost for k -traffic-anonymity.

Finally, our last objective is that the incentives structure of our system encourages full participation of the rational nodes. The critical incentive that we identify is that *participating in a torrent, purely for anonymity reasons, can still offer performance benefits*. This is important for two reasons connected with anonymity. First, to do otherwise would create a system wherein some torrents might only ever have natively interested nodes downloading it. This is a form of anonymity starvation. Second, if there is no value to the cover-traffic torrents in the download set, an inquisitor might be able to distinguish the native-interest in the set. By creating a system where all torrents can be valuable as cover-traffic, nodes have incentives to participate in them preventing torrent starvation and obscuring the native interests of the participants. We emphasize that this is a genuine incentive, requiring no additional enforcement mechanisms or auditing.

In contrast, SwarmScreen is interested in a much weaker attack model. They showed that BitTorrent communities tend to form around *interests* rather than around language, geography, or even friendship. They further showed that these communities can be monitored and classified by observing a small number of the nodes. They describe this invasion of privacy as “guilt by association” attacks. Finally, they also demonstrated that monitoring just 1% of the network is sufficient for assigning users to their communities with 86% accuracy. They solve this attack model by mixing in traffic to other random torrents to obscure which community a SwarmScreen participant belongs to. Defeating this simpler attack model only costs them 25% to 50% overhead.

However, the stronger attack model we defeat with our system is worth the increased cost. An observer that can follow a SwarmScreen node for a long period

of time can easily determine which torrents the node was downloading, because the node never fully downloads the torrents it uses as cover traffic. At the same time, our system also disrupts the guilt by association attack as described.

BitTorrent Anonymity Marketplace, High-Level Design. Our basic system works for any given k level of anonymity. First, each node participates in k different torrents simultaneously. It advertises all k torrents, hereafter called its *active set*, to its local neighborhood. While the composition of the active set can change over time, it must eventually completely download k complete torrents (we will call these the *download set*), or else a long-term observer could immediately filter out the cover-traffic.

Our design also requires that nodes will “cross trade” their torrents, i.e., a node unchoke its peers’ requests for *any* torrent, not just the torrents where a node has benefited from its peers. In our design, a node will consider every possible torrent it sees advertised by its peers, and will prefer to join those torrents which it believes will be most beneficial in its quest to download its native interest.

The design of our valuation function is drawn from models of supply and demand in economics [68]. In general, the value of a torrent to a node is raised by increased numbers of peers that desire it, while the value is lowered by increased numbers of peers that provide it. Unfortunately, it is impossible to directly measure a torrent’s supply and demand in BitTorrent, and so we use several factors to approximate this. These factors include how much of the torrent the peer requires to complete it, *Have* announcements indicating what it is currently trading, and direct *Request* messages to measure what is available.

We highlight that our valuation function was derived from empirical data and

not an economic or mathematical model. Developing a coherent economic valuation function is a significant research undertaking in and of itself and is beyond the scope of this chapter. Our experimental version was constructed by taking the factors that impact the value of a torrent and combining them in a weighted sum. This construction, similar to how utility functions are built [69], enabled us to experiment with different weights for the factors by dialing up or down the constant associated with that variable. Later in this chapter, we will detail our derivation of our constants from experimentation.

The critical hypothesis tested in this work is whether using a valuation function on torrents will drive node behavior such that protocol exchanges related to the native interest are statistically indistinguishable from protocol exchanges for cover traffic. The core idea is that a peer has no idea if a node is asking for pieces of a torrent because it actually wants it, or if it is just asking for those pieces because it has a high value due to the neighborhood's "market" conditions.

4.5 Evaluation

We employed a simulator developed in previous research [70] to evaluate our implementation of the BitTorrent Anonymity Marketplace. The simulator, running faster than real-time, enabled fast design cycles. After completing a simulation, we studied the results, modified the configuration, and re-ran our experiments. This was a significant advantage over using an artificial environment such as PlanetLab or EmuLab to run a "real" BitTorrent client. Simulation is also preferred to releasing a client to public users because it allows us better access to system and client state information and it avoids any potential legal or ethical issues we are not yet prepared to confront.

4.5.1 Implementation

Our client implementation was developed to be as realistic as possible in all stages of their operation. One notable departure from a stock BitTorrent system is that we assume the presence of a distributed hash table (DHT) in which to store metadata, rather than the more limited tracker functionality in the current BitTorrent. What follows is an overview of how nodes participate in the Marketplace.

Publishing. It is essential that objects exchanged in the Marketplace are opaque to users that are uninterested in them. Otherwise, users may choose not to trade in objects they deem overly sensitive. For this reason, all content is encrypted and assigned random identifiers. We assume out-of-band methods (e.g., publisher web servers) help users discover specific torrents and obtain the decryption keys. In this manner, participating nodes will trade in many torrents without any knowledge of their content, except for their own native interest, thus obtaining a modicum of plausible deniability. Once a publisher has encrypted the object and created its random ID, it stores a record similar to a torrent-file into the DHT and announces nodes that are seeding the torrent within the DHT.

Messages. All inter-peer communication consists of unmodified BitTorrent messages with one exception. While normal BitTorrent *Choke* and *Unchoke* messages identify a specific torrent, in the Marketplace these messages are not torrent-specific. These two messages instead signal that the sender is willing or unwilling to fulfill requests for any of the torrents it has currently advertised.

Joining. To use the BitTorrent Anonymity Marketplace, a participant first acquires the random ID for the desired object, as described earlier. Next, the node joins the

DHT and requests a list of active torrents. From this list, the node creates a list of k torrents consisting of its desired torrent plus $k - 1$ randomly selected torrents. The node then indicates to the DHT that it is joining those k torrents and requests participating peers. The node creates a neighborhood from these lists, preferring peers that show up in multiple torrents.

Trading. After nodes join the system, they unchoke peers in a manner similar to BitTorrent with the highest upload services getting the unchoke slots. However, in the Marketplace, all upload service is adjusted by the estimated value of the received pieces. Our implementation keeps the value constant across an entire torrent, although different pieces could ostensibly have different values. Once the values of the upload services are adjusted, unchoking proceeds normally. At the same time, if the node can find a more valuable torrent than the least valuable torrent in its active set, it drops that torrent and joins the new one.

Seeding and Termination. A Marketplace participant must complete k downloads before leaving the system. Before all k torrents have completed, a node may find value in seeding one of its completed torrents, depending on its observations of the supply and demand for those torrents. Alternately, it could forgo seeding and instead look for more profitable ways to trade its available bandwidth.

4.5.2 Development of the Valuation Function

We have developed a valuation function based on reasonable economic assumptions, refined by experimentation, and suitable for enabling our evaluation of our anonymity objectives. We started with basic supply and demand concepts [68]. In other words, we accept the assumption that increased desire and scarcity raise the value of a given

object, while decreased desire and abundance reduce the value of same. In terms of the BitTorrent Anonymity Marketplace, the number of nodes wishing to download a pieces of a torrent constitute the desire, and the nodes that can service those requests constitute the supply. These two factors are the basis for our valuation function.

Unfortunately, the node cannot measure these factors directly and must therefore estimate them. For example, a node sees all the peers within its neighborhood, but it cannot see further. It cannot see every peer participating in every torrent, thus it cannot determine the global supply and demand of torrent pieces, nor even can it determine any other peer's view of this data. To estimate supply, Marketplace nodes treat what they can see, within their own neighborhood, as an estimate for what their peers can see. (Neighborhood visibility is not transitive. If A is in B 's neighborhood and B is in C 's neighborhood, there is no guarantee that A knows anything about C .) Nodes can make a better estimate about the demand for a torrent by totaling the number of pieces required for their peers. They then combine these two estimates into a single factor hereafter referred to as *approximate demand over supply*.

In addition to this information, BitTorrent nodes can make use of the *Have* announcements and *Request* messages from peers to know more about demand in the neighborhood. The *Have* messages indicate a degree of freshness to what torrents neighbors are trading and, of course, *Request* messages are the strongest, most straight-forward measure of demand available.

Our early valuation function was a weighted sum of these three factors. Using this construction, each factor could be experimentally measured to determine if it had an impact at all, and the ideal weighting could be derived experimentally using our simulations. By fixing a weight of 1 to all but one factors, the remaining factor can be evaluated independently. Setting this experimental factor to 0, for example,

completely eliminates its impact on the function.

For testing the Marketplace, we fixed $k = 5$, set the total number of torrents in the marketplace to 40, initiated 100 clients plus 40 seeds, and used 125 MB files for each torrent*. For simplicity, all the clients have the same upload and download speeds of 1Mbps, start at the same time, and end when their k downloads are complete. To test the effects of torrent popularity, we configured 10% of the torrents to be the native interest of 50% of the clients.

Our initial simulations immediately demonstrated that our initial valuation function was insufficient. Regardless of configuration, the clients in the simulation would not complete their downloads. We determined that the nodes were dropping the torrents in their active set, regardless of how much they had completed, for a new torrent that was surging in popularity in their neighborhood. We decreased the frequency at which nodes would update their active set but that didn't solve the problem sufficiently.

After some additional experimentation, we determined that because one of the goals of the node is to complete k downloads, the completeness of a torrent should factor into the valuation function. In other words, if all other factors are equal, a more complete torrent should be valued higher than a less complete one. We retooled the valuation function with this new factor and re-ran the simulations and were rewarded with converging results.

Using our more mature valuation function, we tested the factors in the function independently. For each factor tested, we experimented with weights of 0, 0.25, 0.5, 0.75, 1.0, 2.0, 4.0, 8.0 and 16.0. For completeness, we also tested a few other non-

*Individually 125 MB is a small file for BitTorrent, but because our nodes are exchanging five files simultaneously, the amount of data in transit is 625MB per client.

value related variables such as how often the node updates its active set, and so forth. In total, we ran fifty different configurations of the simulator, again fixing all but one factor at a time and varying it across this broad range of weights.

These tests demonstrated, again, that biases toward completing torrents that have been started are essential, and that data collected from direct requests is the best proxy for overall demand. When we reconfigured the simulation to ignore direct requests, performance worsened by nearly twenty percent. Interestingly, the remaining factors proved to be much poorer estimates of demand and had little impact on average performance. However, they are useful to a node at times when the node has not recently received any such requests. A small weight for these factors was better than no weight at all. We conclude that when the direct request factor is in play, it should dominate the equation. However, when the direct request factor drops to zero, these weaker factors serve as a backup.

While the specific coefficients of valuation function are optimized for our simulation configuration and are thus not directly applicable for a real-world deployment, the insights obtained from this empirical evaluation are still essential. Moreover, we can now test our central hypothesis: will cross-trading nodes that use a valuation function to decide which cover-traffic nodes to trade have the k -traffic-indistinguishability property?

4.5.3 Anonymity Results

To evaluate anonymity, we took the best observed weight for each of the valuation factors and reconfigured the simulator appropriately. With this valuation configuration, we ran twenty simulations. Each took several hours to complete on a 2.4 Ghz Athlon and covered approximately 7 hours of simulation time. Each run involved about

70GB of simulated data transfer and approximately 10 million control messages. The simulations output logs that detail the data transfers and control messages and we used them to trace how the peers interacted with each other as well as to calculate costs and determine performance.

Our primary goal was to quantify indistinguishability of intent. This property means that a node downloading 1 native interest, and $k - 1$ cover traffic torrents will not reveal its native interest by its behavior to its peers. We will examine three node behaviors that could potentially reveal the native interest to peers: start times for torrents, end times for torrents, and download patterns.

Start Time. We first evaluate the indistinguishability of start times, where start time is measured as an integer rank. In other words, the first torrent that a node makes requests for is ranked 1, and the second torrent that a node makes requests for is ranked 2, and so on. We evaluated this aspect of indistinguishability in two ways.

First, we checked that there was sufficient variability of start times for native interests. It is important, of course, that native interests not have a predictable start rank. Our results are shown in Figure 4.1. The graph is parameterized on the number of nodes natively interested in the torrent, as a measure of popularity. The y axis is the average start rank for nodes of that popularity and the standard deviation. The graph shows that the standard deviation is high for start rank, so a node's native interests are suitably obscured from its peers.

Our second measure of the indistinguishability of start times is to measure the average start time for the same torrent for peers that are natively interested relative to peers that are not (see Figure 4.2). There is a noticeable shift to earlier start times for native interests. Nevertheless, the average times for the native interests

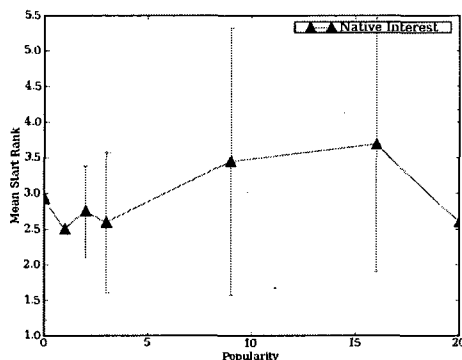


Figure 4.1 : The mean start rank of native interests plotted against popularity. The x -axis is the number of peers natively interested in the torrent, the y -axis is the starting rank. The error bars show the standard deviation. The wide standard deviations mean that native interests have a wide range of start rank.

lie within the standard deviations of the start times for non-native interests. The distributions are not statistically different enough to be detectable. Furthermore, the native and non-native graphs have similar shapes, suggesting similar behavior for the two populations.

End Time. It is also important that native interests not end predictably. Expressing end times as integer ranks, we evaluated the variability of native end times in Figure 4.3 and compared those times to non-native end times in Figure 4.4. These graphs show that, as with start times, there is a wide variability in the end times and that the mean is within the standard deviation of cover-traffic start times.

Download Rates Over Time. Finally, we examined the rate of piece transmissions for native and non-native populations in the Marketplace to verify that transmission *patterns* are indistinguishable. We created our transmission pattern by aggregating each node's download volume within 500 second buckets. All nodes are

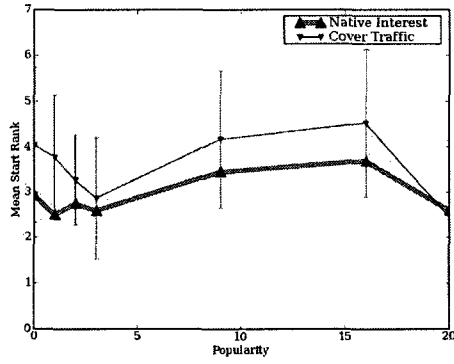


Figure 4.2 : The mean start rank of the various torrents plotted against the start rank for the same torrent for peers not natively interested. This graph shows that native interests do start sooner, but the mean lies within the standard deviation of non-native interest start times for most torrents.

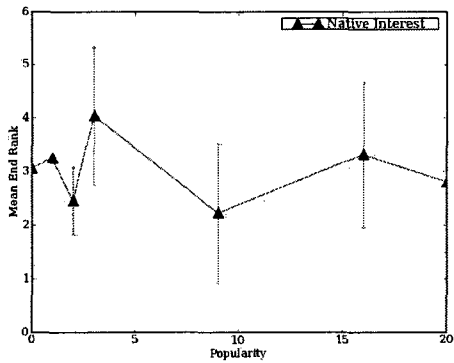


Figure 4.3 : Similar to Figure 4.1, this graph shows the mean ending ranks and the standard deviation. As with start times, end times vary sufficiently to make them poor predictors of interest.

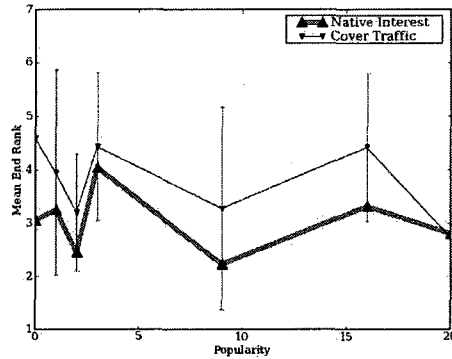


Figure 4.4 : The mean end ranks for native interest compare to mean end ranks for non-native interest. As before, there is a noticeable shift downwards, but, as before, the means for the native interests tend to lie within the standard deviations of the non-native interests.

normalized such that their first 500 second slice of time is slice 0, the second 500 seconds is slice 1, and so forth. Within each slice, we separated the download volume for the native interest from the average download volume for the cover-traffic. The average for all nodes and the standard deviations are computed for each time slice. Figure 4.5 shows the download pattern for all nodes across the entire simulation. We again observe that the nodes' averages for native traffic is within the standard deviation of the cover-traffic. Note also that this graph represents a global view over all nodes, so this any node's local view would have higher error.

We can examine a weaker observer by computing the observed download patterns for a single client. That is, for each node, we aggregated all the traffic that only that node observed directly. As before, we aggregated into 500 second buckets, dividing the native interest traffic from cover traffic. Then we used the average and standard deviations for each node's observed patterns to create Figure 4.6. The two types of traffic overlap even more in this graph, demonstrating that a single peer observes less differences between native interest traffic and cover traffic than can be observed

across the swarm as a whole.

4.5.4 Analysis

We now revisit anonymity against each of the inquisitors that we previously identified.

Passive Inquisitor. These nodes do not directly interact with any actual nodes but only talk to the tracker or DHT. The passive inquisitor can, at best, track a given node's active set. From this information, it cannot determine the node's native interest. As we demonstrated, the entrance and exits of a given torrent in a node's active set appear indistinguishable, regardless of the torrent's status as native interest.

Decoy Passive Inquisitor. These nodes directly interact with other nodes, but do not actually exchange pieces. They can, however, advertise pieces and unchoke other nodes. Such inquisitors gain additional information, because rational nodes will drop them regularly for their poor performance. However, with a Sybil attack [40], these nodes can connect to a given node over and over from different IP addresses, simulating a continuous connection. Such a Sybil attack could track the traffic of a rational node by capturing all *Have* announcements. Nevertheless, even a Sybil attacker will not determine the node's native interest from this information because, as we demonstrated, the download rates for a given torrent for a node are similar, regardless of the node's native or non-native interest in that torrent.

Active Inquisitor. The most powerful non-wiretap node, these nodes actively trade with peers in the network. This feature allows them to attempt to "trick" a victim node into revealing state through carefully crafted trading. For example, an active inquisitor might obtain a large number of blocks from all the nodes in active

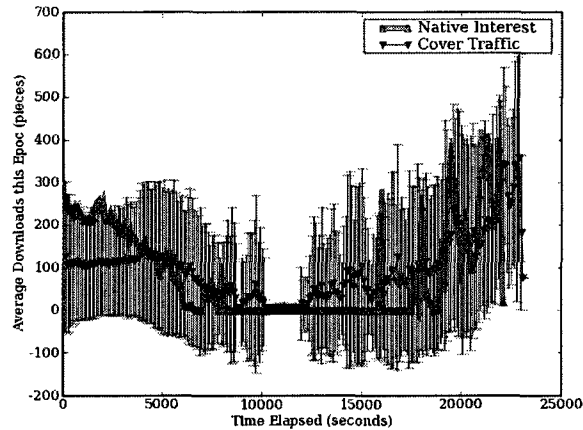


Figure 4.5 : Native and non-native traffic patterns super-imposed. While native traffic is above non-native traffic for the same node, the median for the native is within the standard deviation of the other.

set. Then, it might selectively advertise these blocks to the victim to see which blocks the victim takes a higher interest in. Furthermore, a very well provisioned inquisitor might introduce *identifiable torrents* into the marketplace that it can use to manipulate torrent values within a neighborhood. The active inquisitor can use such value manipulation to attempt to pierce the indistinguishability.

At present, we have not yet attempted to simulate active inquisitors. Nevertheless, we expect that unless the inquisitor can control a large portion of a victim node's local neighborhood (e.g., using a Sybil attack), it cannot have high confidence about the motivation for a node's interest in any given torrent. This attack, however, is made non-trivial because DHTs or trackers give out random subsets of the peers to a participating node, thus dramatically increasing the costs of overtaking a node's neighborhood. Nevertheless, Sybil attacks are a significant security issue and remains a point of research.

In addition to our successful anonymity results, we also quantified the costs in

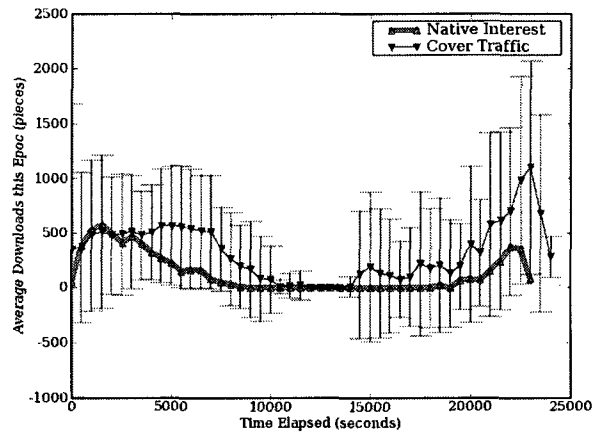


Figure 4.6 : This figure is similar to Figure 4.5 but limited to the viewpoint of single clients. In other words, the former figure is a global representation of download patterns, while this figure is representative of what a single peer observes.

these simulations. The amount of data downloaded, expressed as a multiple of a single torrent, averaged 5.71 ± 0.43 . Given that the optimal value is 5, this indicates that our nodes are not wasting a lot of time downloading torrents that they do not complete.

To conclude our evaluation, we review our incentives qualitatively along two of three axes suggested by previous work [15, 23]. We now consider incentives for communication and incentives for computation. There is no need to evaluate incentives for message passing because the Marketplace, as in regular BitTorrent, does not have peers relay messages for one another.

Incentives for Communication. The first question is, *does a rational node have any incentive to lie about its state?*

1. **Active Torrents:** The only incentive for a node to lie about its active set is for increased anonymity against passive inquisitors. However, we have demon-

strated that the native interest of a node is not revealed by the makeup and dynamics of the active set. Furthermore, the active set is necessary for performance and anonymity. Therefore, there is no incentive to lie about this state.

2. **Choke Status:** There is no incentive for a rational node to misinform a neighbor about the choke state between them. A lie about choke status might result in a snub, which is undesirable.
3. **Piece-Interest Status:** The incentives to lie about this are unclear. There is an incentive for a node to announce that it has pieces, even for pieces it does not actually have because the value of the torrent in the marketplace will increase. On the other hand, unchoked neighbors may ask for these pieces and subsequently snub the lying node when it cannot produce them. We have not yet quantified these incentives, but snubbing is undesirable, providing a disincentive to this behavior.
4. **Piece Requests:** A node has an incentive to request pieces that it already has in order to drive the value of the torrent higher. However, it also affects the value of torrents by pretending not to have the piece. Requesting pieces already present costs additional bandwidth, which is valuable and limited, so that behavior is certainly disincentived. Similarly, pretending not to have a piece means that a peer who might have something to trade might skip over this node. As with core BitTorrent, Marketplace nodes have an incentive to participate normally in torrent trading such that they get what they want in an efficient manner.

Incentives for Computation. We next ask, *does a rational node have any incentive to compute a non-conforming value for torrents in the marketplace?* The answer is no, by definition, because nodes will compute their own market valuations. Theoretically, all nodes have an incentive to develop effective methods of evaluating torrents of non-native interest. The cooperation model supports and encourages this form of self-interested operation.

In summary, the Marketplace is built on a sound foundation of incentives, although some small components are currently manipulable, and aggressive Sybil attacks may be able to weaken the anonymity guarantees. These are open problems for future research.

4.6 Discussion and Future Work

Our proposal of the BitTorrent Anonymity Marketplace is a valuable contribution to p2p-anonymity, particularly if an implementation of it could draw away traffic from Tor. However, our work has produced many more questions than it has answered.

4.6.1 Stronger Anonymity and Ethical Issues

Our anonymity model is designed to shroud a peer's intentions from the observations of its neighbors. However, many BitTorrent users would be interested in shrouding their intentions from adversaries that can tap their wire, such as their ISP. The BitTorrent Anonymity Marketplace could potentially be hardened to improve anonymity in such cases when the adversary can tap the peer's line.

Per-peer encryption. Peers can communicate with one another via encrypted links, an optional feature already present in BitTorrent. This immediately hides the message exchanges that divulge the Marketplace's state. Despite this link encryption,

an adversary would still have access to the public information in the DHT.

Late-start native interest. A node does not need to connect to its native interest upon initialization. Instead, it can choose its k -active set randomly, which may or may not include the native interest. If not present in the initial active set, the node can rotate it into activity at a later time.

Even split peers. Because our system biases a node's selection of its peers based on the value of the torrents they are trading, an observer could approximate the value of each torrent to the node based on its neighbor selection. Nodes could remove this bias, selecting peers evenly from their desired torrents.

Improving cover traffic. Users that are sensitive to their anonymity should ensure that the Marketplace is well stocked with items that are legitimate candidates for cover traffic. Such items would include sensitive, but highly-legal objects that provide better plausible deniability.

This last point about cover traffic leads to interesting ethical questions about the BitTorrent Anonymity Marketplace because it will, without a doubt, provide cover for individuals engaging in illegal and reprehensible behavior. Unfortunately, it is often the assumption that anonymity *only* benefits individuals engaging in such actions. The truth is that anonymity is valuable for many legitimate purposes. For example,

- An individual with a medical condition may not wish to reveal it. Doing research on the internet can expose them to other parties. The BitTorrent Anonymity Marketplace does not provide anonymity for the initial search for documents (a standard service like Tor is well suited to this task), but could provide cover for downloading and viewing a video about treatment options.

- Legality is highly dependent on the jurisdiction. What may be legal in one region of the world may be highly illegal somewhere else. Such content may be sensitive to the downloader even if it is legal. This is especially true if the downloader is from, or has ties to, a jurisdiction where it is illegal.
- Anonymity also protects individuals from commercial exploitation. In cases where BitTorrent is being used for legal content, corporations can easily learn a user's tastes and interests from very simple observations of the tracker or DHT. Absent regulations to the contrary, corporations will naturally begin using this information to target users with advertising and so forth. The BitTorrent Anonymity Marketplace significantly reduces the effectiveness of such attacks, since many or most of the nodes participating in any given torrent will be there for the cover-traffic, not because it's their native interest. In fact, they will have no idea what they're sharing.

The effectiveness of the Marketplace is greatly increases when there are many kinds of legitimate, yet sensitive, torrents actively in trade. On the other hand, if only illegally copied music is found therein, it won't matter if you have k -anonymous cover traffic. K illegal music or movie downloads is no better (and, in fact, could be worse) than just one.

That said, there will be individuals that would be interested in using a service such as the BitTorrent Anonymity Marketplace to engage in illegal behavior. They should be aware that k -traffic anonymity will probably not shield them effectively from government observation (see, e.g., You-are-not-a-lawyer [71]). It is possible, however, that the BitTorrent Anonymity Marketplace *does* help to cover users against corporate investigation. For corporations looking to bring lawsuits against individuals

based on downloads, the BitTorrent Anonymity Marketplace greatly increases the cost of determining infringement, and introduces a risk of false positive to the suing company.

Can a user be held legally liable for downloading a torrent, as cover traffic, assuming the content in question would be illegal to have downloaded via ordinary means? The essence of the user's defense would be that they were just helping random peers to download content, while they, themselves, were getting something entirely different. Of course, if they are faced with all k of their encrypted downloads and asked to prove which one they can decrypt, they may be stuck. Furthermore, even if the user legitimately doesn't know what is being downloaded, the adversary might well crawl the various content discovery sites (e.g., PirateBay and the like), creating their own reverse-mapping from encrypted torrents to their true identities.

As such, the degree of anonymity proffered by the BitTorrent Anonymity Marketplace seems to be comparable to serving as the exit node of Tor or another such onion-route system. The exit node is clearly observable doing fetching what could well be illegal content. The exit node's operator may well claim that the content in question was being delivered to a third party, but the exit node is clearly participating in the process. Of course, such arguments quickly become absurd. Internet core routers certainly have significant volumes of undesirable content transiting them every day, all day long. They might claim a "common carrier" defense if sued. Could a BitTorrent Anonymity Marketplace node, or for that matter a Tor exit node, claim a similar defense?

4.6.2 Informed Risk

One possible development to the BitTorrent Anonymity Marketplace would be channels that inform the participant of risk. In particular, these third-parties would uncover the content names and descriptions for the opaque DHT identifiers. Users of these services could then fill their active sets with elements from white lists or prevent elements from black lists from getting in. This would, of course, erase plausible deniability about not knowing the content. However, the user could choose their own level of risk.

Most importantly users could be absolutely sure that morally, ethically, and legally unacceptably risky content, such as child pornography, would never pass through their systems. Users looking for anonymity for sensitive but legal content, such as medical treatment videos, could also ensure that they were not taking any legal risks for their behavior and might, instead, find themselves downloading medical videos for a wide variety of different ailments. Moreover, certain organizations that believe in civil disobedience to what they perceive as unjust laws might purposefully participate in providing cover traffic for certain classes of torrents. Curiously, the black list for one organization might be a white list for another.

As a concrete example, consider a government that runs a black list of videos that are deemed illegal for whatever reason (e.g., criticism of the king is illegal). Citizens within that country that wish to have anonymity and avoid legal risk could use that list as a black list. Other individuals, inside or outside of the country, might treat that as a white list, looking to provide cover traffic for those torrents by making them more popular.

4.6.3 Future Work

Several aspects of the BitTorrent Anonymity Marketplace remain unresolved or require further exploration. The aforementioned legal issues are one such area. It would be valuable to explore the legal possibilities of the BitTorrent Anonymity Marketplace under the laws of various jurisdictions.

Another area of significant future research is the valuation function that each peer performs on the torrents it is trading. Just as we are not lawyers, we are also not economists. We recognize that the economic interactions of our proposed system are complicated but subtle. In a real world implementation, there might be thousands of torrents and hundreds of thousands of clients in the Marketplace, not to mention churn, disparities of upload and download capacities and so forth. It will be a daunting challenge to uncover a generalized valuation function that works well under all circumstances.

Our current simulations are pedagogical and unrealistic. In particular, we have not studied the BitTorrent Anonymity Marketplace under realistic churn or other such conditions. Because our simulations lack these features, we have been unable to see some predicted behaviors that require them. Also, in a real-world scenario, torrents will be of different sizes and nodes would have widely varying network performance. Different nodes might have different values of k -anonymity that they desire. It would be convenient if the choice of k value for a client had no impact on its neighbors, but we have not examined this.

We have also not completely explored the attack space for either inquisitors or rational attackers. Our simulation does not yet include an active inquisitor that attempts to introduce tainted information in an effort to reveal the interests of peers. Similarly, our simulations do not yet include a rational manipulators that lies about

state in an effort to manipulate torrent values.

Finally, it should be obvious that simulation alone is insufficient for evaluating the BitTorrent Anonymity Marketplace. An actual implementation must be created and evaluated for real-world operations. A whole host of difficulties is involved in such development, although most of them are legal, rather than technical.

4.7 Conclusion

In this work, we have explored a new method for cooperative anonymity in BitTorrent swarms, called the BitTorrent Anonymity Marketplace, where peers exchange pieces of multiple torrents based on their value for trading with other peers. This creates a world where intent is difficult to discern because motivations are obscured by the shifting values within the local neighborhood. Nodes always download k different torrents, selected randomly, to completion, obscuring their true intent, yet still biased in favor of increasing the nodes' observed performance.

With detailed event-based simulations, we demonstrated that the download behavior for native interests and cover traffic was statistically similar, making it difficult for observers to distinguish between the two. We also demonstrated in simulation that our Marketplace completes without unreasonable overhead beyond the cover traffic's costs. We also evaluated the incentives of our system and found that the overall setup is sound against rational manipulations, but that there are obvious places for exploitation.

Chapter 5

Conclusions

In this thesis, we have investigated one of the most difficult problems for peer-to-peer networks: preventing rational attacks through proper incentives design. Our results contribute to both the theory and practice of such design by elucidating attributes important to successful incentives and then using those principles in concrete implementations. Specifically, genuine incentives generally provide more effective rational robustness than their artificial counterparts. In fact, genuine incentives completely eliminate the *auditing* class of attacks that many artificial-incentives systems are subject to. Armed with this key observation, we implemented two extensions to the BitTorrent p2p protocol that improve performance and anonymity. Both extensions have the incentives built directly into the peer-interactions requiring no additional infrastructure for behavior enforcement.

In this chapter we present our conclusions regarding genuine incentives given our experiences developing the two aforementioned extensions. This discussion excludes any analysis of the effectiveness of the extensions in achieving their designated objectives. Such analysis is already provided in their respective individual chapters. Instead, our conclusions in this final chapter are focused on how our understanding of genuine incentives has grown through the development of these extensions.

In summary, our experiences have strengthened our belief in the superiority of genuine incentives. We observed them to be simple in design and impervious to auditing attacks by definition. Those observations were not novel, but were reassuring. On the

other hand, in the development of our extensions, we also observed that a genuine incentives system was also easy to understand from a rational agent's perspective. In other words, the incentives system was so simple that the agent could easily be expected to understand the mechanism. This is critical to rational robustness as we will describe later in the chapter.

5.1 Genuine Incentives: Simplicity

When we began to approach the problem of adding incentives for seeding in BitTorrent, our first solution used an artificial incentives. The basic idea was for a node to seed in return for a cryptographically signed token from the receiver. In the future, if the nodes were to re-encounter one another, the previous seeder could "demand" service from the previous receiver by returning the tokens. Failure to honor these commitments would result in a negative reputation. We then began designing the system to allow for the trading of these tokens between peers to create a form of virtual currency.

While the design is interesting, we quickly abandoned it for the sheer complexity of the mechanism. Consider these problems:

1. *Cross Trading Value* - If cross trading of tokens is supported, so that a node trades tokens with another node, the purchasing node must have some belief that the tokens will be honored. Otherwise, they would have no value. This means that the purchaser must expect to encounter the tokens' signer in the future, or another node that is willing to purchase them. This would obviously be a probabilistic computation, but we could not easily identify factors that would influence the probabilities.
2. *Reputation* - To adequately punish nodes that refused to honor their signed

tokens, the slighted node would have to convince peers to refuse to interact with the transgressor. This is complicated for two reasons. First, the method for accusing a node of breaking its word must deal with problems such as false accusations, and a number of other difficult issues. Second, the accuser must be able to reach a large number of peers in the BitTorrent swarm to be able to have any effect. In any event, even if the accuser can prove that another node is slighting it, other peers may choose to do business with it anyway because it has always been honest with them.

This solution was discarded because of its overwhelming complexity in favor of the solution we implemented. While our genuine incentives approach has some non-trivial design concerns, the overall system is significantly simpler.

Obviously, this is merely an anecdotal evidence, but it does illustrate typical problems encountered by artificial incentives. In general, artificial incentives require an additional enforcement protocol in addition to the cooperative protocol. This lack of protocol cohesion introduces extra complexity that makes implementation difficult and also more prone to rational attacks simply because the attack surface is larger.

This is not to say that creating genuine incentives is easy. As we will describe later, genuine incentives require greater design time than artificial ones.

5.2 Genuine Incentives: Impervious to Auditing Attacks

The enforcement protocol common to artificial incentives system is often designed around an auditing mechanism of some type. In the original design we suggested for seeding incentives, the tokens issued by a node were cryptographically signed to be used as a form of evidence. In theory, unhonored tokens would be shown to other peers that would subsequently shun the offender.

Unfortunately, this relatively simple idea is complicated in practice. Consider a malicious node that falsely accuses a victim of refusing to honor its tokens. A properly designed crypto system provides a mechanism for the victim to refute the charges, but the victim is unable to trace all of the gossip. This means that nodes hearing the accusations would either have to trust them, or validate them directly. The latter option is costly because they must either stop their current operations to track down the accused node to determine guilt, or wait until they encounter the node in normal interactions then ask for refutation of all accusations received until that point.

The problem with auditing attacks drives to the heart of the problem of artificial incentives. If the primary protocol can proceed without the enforcement protocol, then a rational attacker need only disable or disrupt the latter to misuse the former.

On the other hand, genuine incentives generally forgo auditing altogether and this entire attack class disappears. Most systems of this type rely on first-hand information, throwing away all the complications of trust, reputation, and third-party enforcement. Obviously, this reduction also introduces limitations that we will discuss later.

5.3 Genuine Incentives: Bounded Rationality

In our discussion thus far, we have modeled P2P nodes as rational agents. That means they know the rules of the game, and will use those rules to maximize their utility.

In real life, however, most participants are not rational by these definitions, but are *bounded rational*. Bounded rational parties want to maximize their utility, but either do not know the full rules of the game, or have insufficient resources to play the maximizing strategy. Restated, even if an incentives system is designed such that

obedience is the utility maximizing strategy, participants may choose a different strategy because they do not know better or because obedience requires more resources than they have available.

Consider for example an artificial incentives system with an auditing component. Even if the auditing mechanism is impervious to attack, if the system is sufficiently complicated, rational attackers may *believe* that the mechanism can be circumvented. The result may be that the attackers play sub-optimal strategies to their own detriment, but also to the detriment of the entire system.

Genuine incentives, however, are simple and easy-to-understand and difficult or expensive to abuse. For example, it has been shown that BitTorrent is imminently abusable, yet it has been one of the most stable p2p systems to date. Despite all of the known methods for exploiting BitTorrent, enough nodes continue to cooperate to keep the system viable. We believe this is partially because the trading mechanism is easy to understand and easy to follow, while the exploits are complicated to understand or have a high barrier for entry such as a non-standard or unpopular client.

Bounded rationality is probably the strongest reason to choose genuine over artificial incentives. Even if artificial incentives are robust and stable, their complexity may still result in bounded rational participants being disobedient. Conversely, a genuine incentive with provable vulnerabilities might still be more effective if the incentive is easy and vulnerabilities are hard.

5.4 Genuine Incentives: Limitations

Despite the various advantages of genuine incentives, we have identified two limitations designers must account for.

First, genuine incentives require a greater design time. While they are generally

more simple in implementation, the incentive itself is often difficult to identify. Creating a genuine incentive requires tying incentives and resource-exchange together. The implementation of such an entanglement is relatively simple, but discovering the entanglement is difficult and time consuming.

Second, there is no free lunch. Genuine incentives cannot be optimal for all possible p2p problems. They are, by definition, limited to influencing behaviors tied to resource exchange. While this may sound obvious, consider the following two common uses of artificial incentives for which there are no genuine equivalents. First, *artificial incentives can be, and are often, used to enable transitive cooperation*. This enables nodes that have not encountered one another before to not have to start from scratch in their cooperative relationship because of some transitive relationship. Second, punishments from artificial incentives can include the excommunication of the offender from the swarm. This means that a wider variety of bad behaviors can be penalized because the penalty is not just tied to performance.

These were issues we dealt with in the process of developing our own extensions. Because we made the conscious choice of exclusively using genuine incentives, we had to design around these limitations.

5.5 Genuine Incentives: Future Work and the Final Word

The concept of a genuine versus artificial incentive is entirely novel and an important contribution to p2p design theory. In this work we have investigated the nature of genuine incentives through survey and experimentation. In doing so, we have illuminated the concept and demonstrated that it is a useful design principle. We feel this justifies future research and propose the following questions as starting points.

Can the definition of genuine incentives be formalized? We have used a loose

and informal definition of genuine and artificial incentives throughout this work. It has been useful as a design principle and a method of classifying systems. Moreover, it has helped provide useful insight. Nevertheless, the next step is to formalize its definition and clearly identify its characteristics and limitations.

Can we formally prove that genuine incentives are more robust than artificial ones? Our assertions about the value of genuine incentives are based on experience, surveys, and reasoning. Obviously, any incentive that does not use auditing is not subject to auditing attacks, but this is not the salient point. If we succeeded in creating a formal definition of genuine incentive, then perhaps we could subsequently investigate a formal model of its robustness.

How can artificial incentives be designed to bolster genuine incentives? Given that all behaviors cannot be genuinely incentivized, artificial incentives may often need to be added to a genuine incentives system. However, it is not clear if a p2p system would benefit from the two systems designed to inter-operate as opposed to simply co-existing.

Breaking the limitations of genuine incentives Although we found it difficult to use genuine incentives beyond direct-contact peers, we have no proof that such incentives do not exist. Genuine incentives of this form could greatly expand the robustness of many p2p systems. Such a solution would have to tie the transfer of cooperation into the direct transfer of resources. We also identified excommunication of a node as something tied to artificial incentives. Uncovering a genuine incentives method of kicking a node out of a system would also be beneficial to p2p systems.

Having identified these avenues for further investigation, we conclude this thesis by restating our fundamental contribution. In this work we have identified an important class of incentives in p2p systems: *genuine incentives*. In surveys of existing

systems we identified the strength of these incentives versus their artificial counterparts. We then used this observation to create two extensions to BitTorrent based on genuine incentives. In this endeavor, we strengthened our understanding of how this type of incentive works, as well as what its limitations are. More importantly, we demonstrated that this concept was useful in the design of practical p2p systems. Our extensions solved unrelated problems, but in both cases, the guiding design goal was to use a genuine incentive and the results were effective and interesting. We hope that this research aides in the development of future p2p computing by improving incentives design.

Bibliography

- [1] B. Cohen, "Incentives build robustness in BitTorrent," in *Proceedings of the 1st International Workshop on Economics of P2P Systems (P2PECON '03)*, (Berkeley, CA), June 2003.
- [2] D. R. Sandler and D. S. Wallach, "Birds of a fethr: Open, decentralized micropublishing," in *Proceedings of the 8th International Workshop on Peer-to-Peer Systems (IPTPS '09)*, (Boston, MA), 2009.
- [3] X. Yang, M. Gjoka, P. Chhabra, A. Markopoulou, and P. Radriguez, "Kangaroo: Video seeking in p2p systems," in *Proceedings of the 8th International Workshop on Peer-to-Peer Systems (IPTPS '09)*, (Boston, MA), 2009.
- [4] J. Terrace, H. Laidlaw, H. E. Liu, S. Stern, and M. J. Freedman, "Bringing p2p to the web: Security and privacy in the firecoral network," in *Proceedings of the 8th International Workshop on Peer-to-Peer Systems (IPTPS '09)*, (Boston, MA), 2009.

- [5] A. Post, P. Kuznetsov, and P. Druschel, "Podbase: transparent storage management for personal devices," in *Proceedings of the 7th International Workshop on Peer-to-Peer Systems (IPTPS '08)*, (Tampa Bay, FL), 2008.
- [6] S. Guha, N. Daswani, and R. Jain, "An experimental study of the skype peer-to-peer VoIP system," in *Proceedings of the 6th International Workshop on Peer-to-Peer Systems (IPTPS '04)*, 2007.
- [7] F. Kaashoek and D. R. Karger, "Koorde: A simple degree-optimal hash table," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, (Berkeley, CA), 2003.
- [8] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware '01)*, (Heidelberg, Germany), Nov. 2001.
- [9] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-resilient wide-area location and routing," Tech. Rep. UCB-CSD-01-1141, U. C. Berkeley, Apr. 2001.
- [10] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications (ACM SIGCOMM '01)*, (San Diego, CA), Aug. 2001.
- [11] E. Adar and B. A. Huberman, "Free riding on gnutella," *First Monday*, vol. 5, 2000.

- [12] L. Ramaswamy, “Free riding: A new challenge to peer-to-peer file sharing systems,” in *In Proceedings of Hawaii International Conference on Systems Science* 36, 2003.
- [13] M. Feldman, C. Papadimitriou, and J. Chuang, “Free-riding and whitewashing in peer-to-peer systems,” in *In Proc. PINS*, pp. 228–236, 2004.
- [14] F. Liu, B. Li, and L. Zhong, “How p2p streaming systems scale over time under a flash crowd,” in *Proceedings of the 8th International Workshop on Peer-to-Peer Systems (IPTPS '09)*, (Boston, MA), 2009.
- [15] J. Shneidman and D. C. Parkes, “Specification faithfulness in networks with rational nodes,” in *Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing (PODC'04)*, (St. John's, Canada), July 2004.
- [16] K. Leyton-brown, I. Mironov, and M. Lillibridge, “Incentives for sharing in peer-to-peer networks,” in *In Proceedings of the 3rd ACM conference on Electronic Commerce*, 2001.
- [17] K. Lai, M. Feldman, I. Stoica, and J. Chuang, “Incentives for cooperation in peer-to-peer networks,” in *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems (P2PECON '03)*, (Berkeley, CA), 2003.
- [18] T.-W. J. Ngan, D. S. Wallach, and P. Druschel, “Incentives-compatible peer-to-peer multicast,” in *Proceedings of the 2nd Workshop on the Economics of Peer-to-Peer Systems (P2PECON '04)*, (Cambridge, MA), June 2004.
- [19] M. Feldman, K. Lai, I. Stoica, and J. Chuang, “Robust incentive techniques for peer-to-peer networks,” in *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, (New York, NY), pp. 102–111, 2004.

- [20] M. Roussopoulos, M. Baker, and D. S. H. Rosenthal, “2 p2p or not 2 p2p?,” in *Proceedings of the Third International Workshop on Peer-to-Peer Systems (IPTPS '04)*, Feb. 2004.
- [21] J. Shneidman and D. Parkes, “Rationality and self-interest in peer to peer networks,” in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, (Berkeley, CA, USA), Feb. 2003.
- [22] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, “Secure routing for structured peer-to-peer overlay networks,” in *Proceedings of the 5th Symposium on Operating System Design and Implementation (OSDI '02)*, (Boston, MA), Dec. 2002.
- [23] J. Shneidman, D. C. Parkes, and L. Massoulié, “Faithfulness in internet algorithms,” in *Proceedings of the SIGCOMM Workshop on Practice and Theory of Incentives and Game Theory in Networked Systems (PINS'04)*, (Portland, OR, USA), Sept. 2004.
- [24] R. Axelrod and W. D. Hamilton, “The evolution of cooperation,” *Science*, vol. 211, pp. 1390–1396, 1981.
- [25] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, “Do incentives build robustness in BitTorrent?,” in *Proceedings of 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI 2007)*, (Cambridge, MA), April 2007.
- [26] Y. Tian, D. Wu, and K. W. Ng, “Modeling, analysis and improvement for BitTorrent-like file sharing networks,” in *Proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, (Barcelona,

- Spain), Apr. 2006.
- [27] M. Sirivianos, J. H. Park, R. Chen, and X. Yang, “Free-riding in BitTorrent with the large view exploit,” in *Proceedings of the 6th International Workshop on Peer-to-Peer Systems (IPTPS '07)*, (Bellevue, WA), 2007.
- [28] A. Legout, N. Liogkas, E. Kohler, and L. Zhang, “Clustering and sharing incentives in BitTorrent systems,” in *Proceedings of the 2007 ACM SIGMETRICS*, (San Diego, CA), pp. 301–312, June 2007.
- [29] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer, “Free riding in BitTorrent is cheap,” in *Proceedings of the 5th Workshop on Hot Topics in Networks (HotNets-V)*, (Irvine, CA), Nov. 2006.
- [30] A. Parker, *The True Picture of Peer-to-Peer Filesharing*. CacheLogic, 2004. No longer available from original site http://www.cachelogic.com/home/pages/studies/2004_01.php. Downloaded from archive.org on January 8, 2009.
- [31] Ernesto, “BitTorrent: The “one third of all internet traffic” myth,” Sept. 2006. Viewed at <http://torrentfreak.com/bittorrent-the-one-third-of-all-internet-traffic-myth/> on January 8, 2009.
- [32] R. Singel, *Internet Mysteries: How Much File Sharing Traffic Travels the Net?* Wired, May 2008. Viewed at <http://blog.wired.com/27bstroke6/2008/05/how-much-file-s.html> on January 8, 2009.
- [33] J. Pouwelse, P. Garbacki, D. Epema, and H. J. Sips, “The BitTorrent P2P file-sharing system: Measurements and analysis,” in *4th International Workshop on*

- Peer-to-Peer Systems (IPTPS '05)*, vol. 3640, (Ithaca, NY), pp. 205–216, Feb. 2005.
- [34] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The second-generation onion router,” in *Proceedings of the 13th USENIX Security Symposium*, (San Diego, CA), August 2004.
- [35] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, “Shining light in dark places: Understanding the tor network,” in *Proceedings of the 8th Privacy Enhancing Technologies Symposium (PETS 2008)*, (Leuven, Belgium), July 2008.
- [36] K. Bauer, D. McCoy, D. Grunwald, and D. Sicker, “Bitblender: Light-weight anonymity for BitTorrent,” in *Proceedings of the Workshop on Applications of Private and Anonymous Communications (AIPACa 2008) in conjunction with SecureComm 2008*, (Istanbul, Turkey), September 2008.
- [37] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson, “Friend-to-friend data sharing with oneswarm,” Tech. Rep. N/A, UW-CSE, February 2009.
- [38] D. Choffnes, J. Duch, D. Malmgren, R. Guimer, L. Amaral, and F. E. Bustamante, “Swarmscreen: Pirvacy through plausible deniability in p2p systems,” Tech. Rep. N/A, Northwestern EECS, March 2009.
- [39] S. Glassman, M. Manasse, M. Abadi, P. Gauthier, and P. Sobalvarro, “The militant protocol for inexpensive electronic commerce,” *World Wide Web Journal, Fourth International World Wide Web Conference Proceedings*, vol. 1, no. 1, pp. 603–618, 1996.

- [40] J. R. Douceur, “The Sybil attack,” in *Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, (Cambridge, Massachusetts), Mar. 2002.
- [41] “Microsoft “Palladium”: A business overview,” 2002. <http://www.microsoft.com/presspass/features/2002/jul02/0724palladiumwp.asp>.
- [42] TCPA, “Building a foundation of trust in the PC,” tech. rep., Trusted Computing Platform Alliance, 2000.
- [43] “‘Trusted Computing’ frequently asked questions,” 2003. <http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html>.
- [44] L. P. Cox and B. D. Noble, “Samsara: Honor among thieves in peer-to-peer storage,” in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (SOSP '03)*, pp. 120–132, ACM Press, Oct. 2003.
- [45] M. Waldman and D. Mazieres, “Tangler: a censorship-resistant publishing system based on document entanglements,” in *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pp. 126–135, ACM Press, Nov. 2001.
- [46] E. J. Friedman and P. Resnick, “The social cost of cheap pseudonyms,” *Journal of Economics & Management Strategy*, vol. 10, pp. 173–199, June 2001.
- [47] A. C. Fuqua, T.-W. J. Ngan, and D. S. Wallach, “Economic behavior of peer-to-peer storage networks,” in *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems (P2PECON '03)*, (Berkeley, CA), June 2003.

- [48] W. A. Arbaugh, D. J. Farber, and J. M. Smith, "A secure and reliable bootstrap architecture," in *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, (San Diego, CA, USA), p. 65, IEEE Computer Society, May 1997.
- [49] T.-W. J. Ngan, A. Nandi, A. Singh, D. S. Wallach, and P. Druschel, "On designing incentives-compatible peer-to-peer systems," in *2nd Bertinoro Workshop on Future Directions in Distributed Computing (FuDiCo II: S.O.S.)*, (Bertinoro, Italy), June 2004.
- [50] T.-W. J. Ngan, D. S. Wallach, and P. Druschel, "Enforcing fair sharing of peer-to-peer resources," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), LNCS 2735*, (Berkeley, CA), pp. 149–159, Feb. 2003.
- [51] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th International Conference on World Wide Web*, pp. 640–651, May 2003.
- [52] P. Maniatis and M. Baker, "Secure history preservation through timeline entanglement," in *Proceedings of the 11th USENIX Security Symposium*, pp. 297–312, USENIX Association, 2002.
- [53] H. Schulze and K. Mochalski, *Internet Study 2007*. Ipoque, 2007. Downloaded at http://www.ipoque.com/userfiles/file/internet_study_2007.pdf on January 8, 2009.
- [54] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, and A. Iosup, "Tribler: A social-based peer-to-peer system," in *Proceedings of the 5th International P2P conference (IPTPS '06)*, (Santa Barbara, CA), Feb. 2006.

- [55] C. A. Waldspurger and W. E. Weihl, “Lottery scheduling: flexible proportional-share resource management,” in *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation (OSDI '94)*, (Monterey, CA), Nov. 1994.
- [56] S. Nielson, S. Crosby, and D. Wallach, “A taxonomy of rational attacks,” in *The 4th Annual International Workshop on Peer-To-Peer Systems (IPTPS '05)*, (Ithaca, NY), Feb. 2005.
- [57] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, “Analyzing and improving BitTorrent performance,” Tech. Rep. MSR-TR-2005-03, Microsoft Research, Redmond, WA, Feb. 2005.
- [58] K. Eger, T. Hoßfeld, A. Binzenhöfer, and G. Kunzmann, “Efficient simulation of large-scale P2P networks: Packet-level vs. flow-level simulations,” in *2nd Workshop on the Use of P2P, GRID and Agents for the Development of Content Networks (UPGRADE-CN'07)*, (Monterey, CA), pp. 9–16, June 2007.
- [59] W. Yang and N. Abu-Ghazaleh, “GPS: a general peer-to-peer simulator and its use for modeling BitTorrent,” in *13th Annual Meeting of the IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2007)*, (Atlanta, GA), pp. 425–432, Oct. 2005.
- [60] D. Carra, G. Neglia, and P. Michiardi, “On the impact of greedy strategies in BitTorrent networks: The case of BitTyrant,” in *Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing (P2P '08)*, (Aachen, Germany), pp. 311–320, Sept. 2008.

- [61] A. Nandi, T.-W. J. Ngan, A. Singh, P. Druschel, and D. S. Wallach, "Scrivener: Providing incentives in cooperative content distribution systems," in *ACM/IFIP/USENIX 6th International Middleware Conference (Middleware 2005)*, (Grenoble, France), Nov. 2005.
- [62] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 367–378, 2004.
- [63] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest first and choke algorithms are enough," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC '06)*, (Rio de Janeiro, Brazil), pp. 203–216, Oct. 2006.
- [64] K. D. Vogeleer, D. Erman, and A. Popescu, "Simulating BitTorrent," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems Workshop (SIMUTools '08)*, (Marseille, France), pp. 1–7, Mar. 2008.
- [65] S. Jun and M. Ahamad, "Incentives in BitTorrent induce free riding," in *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems*, (Philadelphia, PA), pp. 116–121, 2005.
- [66] T.-W. J. Ngan, R. Dingleline, and D. S. Wallach, "Building incentives into tor," Tech. Rep. TR08-09, Rice University, Department of Computer Science, 2008.
- [67] A. Androulaki, M. Raykova, S. Srivatsan, A. Stavrou, and S. M. Bellovin, "Par: Payment for anonymous routing," in *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, (Leuven, Belgium), July 2008.

- [68] A. Smith, *An Inquiry into the Nature and Causes of the Wealth of Nations*. London, England: W. Strahan and T. Cadel, fifth edition ed., 1776.
- [69] P. K. Dutta, *Strategies and Games*, ch. Utility and Expected Utility. Massachusetts Institute of Technology, 2001.
- [70] S. J. Nielson and D. S. Wallach, “Building better incentives for robustness in bittorrent.” http://sys.cs.rice.edu/~sethn/btsim/output/longterm_incentives.pdf, January 2009.
- [71] O. Paul, “Being acquitted versus being searched (yanal).” <http://www.freedom-to-tinker.com/blog/paul/being-acquitted-versus-being-searched-yanal>, February 2009.