INVESTIGATING AND IMPROVING BITTORRENT'S PIECE AND NEIGHBOR SELECTION ALGORITHMS

by

Cameron Dale
B.Sc., Simon Fraser University, 2000

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

APPROVAL

Name:	Cameron Dale
Degree:	Master of Science
Title of thesis:	Investigating and Improving BitTorrent's Piece and Neighbor Selection Algorithms

Examining Committee: Greg Baker
Chair

Dr. Jiangchuan Liu, Senior Supervisor

Dr. Joseph Peters, Supervisor

Dr. Art Liestman, Supervisor

Dr. Qianping Gu, SFU Examiner

Date Approved: ______________________

Abstract

In this thesis, we examine two important factors in the design of BitTorrent: how it chooses pieces and neighbors. We present a measurement study on the distribution and evolution of the pieces in BitTorrent, from data collected by multiple administered clients distributed in different parts of the network. Our results validate that the downloading policy of BitTorrent is effective, yet enhancements are still possible to achieve the ideal piece distribution. We also consider the topologies of multiple complex networks formed by neighbor selection in BitTorrent. Our results demonstrate that the networks exhibit fundamental differences during different stages of a swarm, and we discover the presence of a robust scale-free network in the network of peer unchokings. However, unlike previous studies, we find no evidence of persistent clustering in any of the networks. We therefore present a first attempt to introduce clustering, and verify its effectiveness through simulations and experiments.

Keywords:

BitTorrent; P2P; peer-to-peer; network topology; small-world; scale-free

Subject Terms:

Peer-to-peer architecture (Computer networks); Electric network topology; Network performance (telecommunications); Communications network architecture

Acknowledgments

I would first like to thank my senior supervisor Dr. Jiangchuan Liu. He was a source of guidance and inspiration throughout my graduate work. His support of my research was unfailing and consistently exceeded my expectations, and without which this thesis would not have been possible.

I would also like to thank my supervisor Dr. Joseph Peters. His teachings, both in and out of the classroom, and the discussions I have had with him on the areas of this thesis and others have enriched my graduate experience.

I also thank all the members of my examining committee for reviewing my thesis, Dr. Art Liestman and Dr. Mohammed Hefeeda for taking me on when I was an untried graduate student just accepted to the university, Greg Baker for taking the time to chair my thesis defense, Dr. Qianping Gu for sitting as examiner on my examining committee, and all the other faculty and staff in the School of Computing Science at Simon Fraser University who I have interacted with in the last 2 years.

I thank my colleagues and friends at Simon Fraser University for their help: Xu Cheng for our work together on a paper and help with preparing this thesis, Dr. Dan Wang for his work and correspondence on some theoretical issues, Michael Letourneau for his endless support with some of the hardware used in this research, and all others for their assistance with, and encouragement of, my research.

And finally, I thank my friends and family for their support and encouragement in completing this accomplishment. I would especially like to thank Syama Chatterton for allowing me to draw on her invaluable experience as a graduate student.

Contents

Approval **ii**

Abstract **iii**

Acknowledgments **iv**

Contents **v**

List of Tables **viii**

List of Figures **ix**

1 Introduction **1**

1.1 Piece Selection 1

1.2 Neighbor Selection 2

1.3 Organization of the Work 3

2 Related Work **4**

2.1 Piece Selection 4

2.2 Neighbor Selection 5

3 Background **6**

3.1 BitTorrent 6

3.1.1 Definitions 6

3.1.2 The Rarest-First Policy 7

3.1.3 Piece Population 8

3.1.4 Block Variation 9

3.1.5 The Incentive Mechanism 9
3.1.6 Networks in BitTorrent Swarms 10
3.2 Network Topologies 11
3.2.1 Scale-Free Networks 11
3.2.2 Small-World Networks 13

4 Experimental Design 15
4.1 Running BitTornado on PlanetLab 15
4.2 Piece Population 17
4.2.1 Real Internet Swarms 17
4.2.2 Simulated PlanetLab Swarms 18
4.3 Network Topologies 18

5 Piece Population 20
5.1 Snapshots 20
5.2 Evolution 22
5.2.1 The feisty Swarm 22
5.2.2 The openSUSE-2 Swarm 27
5.3 Simulated PlanetLab Swarms 31
5.3.1 The PlanetLab-1 Swarm 31
5.3.2 The PlanetLab-2 Swarm 36

6 Network Topologies 41
6.1 Characteristics of Network Topologies 41
6.1.1 Scale-Free Characteristics 42
6.1.2 Small-World Characteristics 44
6.2 Connectivity Matrix 47
6.3 Impact of Churn 50

7 Small-World Enhancement 53
7.1 Theory 53
7.1.1 Maximum Clustering Coefficient 54
7.1.2 Expanding Diameter 55
7.1.3 Forming the Small-World 56

7.2 Implementation 56
7.2.1 Tracker Modification 56
7.2.2 Simulation Results 59
7.2.3 Experimental Results 60

8 Discussion **67**
8.1 Piece Population 67
8.1.1 Future Work 68
8.2 Network Topologies 69
8.2.1 Future Work 70

9 Conclusions **72**

Reference List **73**

List of Tables

4.1 The distribution of peer characteristics used in the simulated BitTorrent swarms. 16
4.2 The torrents used for the piece population experiments. 17

5.1 The block variation for the snapshot measurements of real internet swarms. . 22

List of Figures

3.1 A power-law distribution of node degrees. 12
3.2 The neighborhood of a single node with a clustering coefficient of 0.6. 13

5.1 The piece population snapshots of four real Internet swarms. 21
5.2 The number of peers in the feisty swarm. 23
5.3 Selected piece populations from the feisty swarm. 24
5.4 The mean of the piece population in the feisty swarm. 25
5.5 The block variation of the piece population in the feisty swarm. 26
5.6 The number of peers in the openSUSE-2 swarm. 27
5.7 Selected piece populations from the openSUSE-2 swarm. 28
5.8 The mean of the piece population in the openSUSE-2 swarm. 29
5.9 The block variation of the piece population in the openSUSE-2 swarm. 30
5.10 The number of peers in the PlanetLab-1 swarm. 32
5.11 Selected piece populations from the PlanetLab-1 swarm. 33
5.12 The mean of the piece population in the PlanetLab-1 swarm 34
5.13 The block variation of the piece population in the PlanetLab-1 swarm. 35
5.14 The number of peers in the PlanetLab-2 swarm. 37
5.15 The mean of the piece population in the PlanetLab-2 swarm 38
5.16 The block variation of the piece population in the PlanetLab-2 swarm. 39
5.17 The block variation of the piece population using a random piece selection algorithm. 40

6.1 The population of peers in the system during the first 45 hours of the experiments. 42

6.2 The node in-degree distribution for the Unchoked network at hour 19 of the experiments, and the resulting fit to it. 43
6.3 The goodness of fit (R^2) parameter measured during the experiments. 44
6.4 The exponent found by fitting a power law to the Unchoked graph's node degree during the experiments. 45
6.5 The characteristic path length during the experiments. 46
6.6 The clustering coefficient during the experiments. 47
6.7 The clustering coefficient during the experiments compared with the clustering coefficient of a similar random graph. 48
6.8 The connectivity matrix at hour 4. 49
6.9 The connectivity matrix from the previous results by Al-Hamra et. al. 50
6.10 The connectivity matrix at hour 8. 51
6.11 The connectivity matrix at hour 16. 51
6.12 The connectivity matrix at hour 32. 51
6.13 The clustering coefficient during the experiment with varying churn. 52

7.1 An example of one of the 5-cliques from a cycle. 54
7.2 The change in the clustering coefficients and characteristic path lengths of the n-clique graphs with varying amounts of randomness added. 57
7.3 The simulated connectivity matrix after 400 peers have joined the swarm. . . 58
7.4 The simulated small-world tracker results for three sizes of graphs. 59
7.5 The simulated small-world tracker results for three values of the `clique-size` configuration parameter. 61
7.6 The characteristic path lengths during the small-world tracker experiment. . 62
7.7 The clustering coefficients during the small-world tracker experiment. 63
7.8 The clustering coefficients during the small-world tracker experiment, compared with the clustering coefficient of a similar random graph. 64
7.9 The connectivity matrix at hour 4 of the small-world tracker experiment. . . 65
7.10 The connectivity matrix at hour 32 of the small-world tracker experiment. . . 66

Chapter 1

Introduction

Among all the peer-to-peer Internet applications available, BitTorrent [10] has become the most popular for the sharing of large files. Recent reports have indicated that half of all the current Internet traffic is due to BitTorrent [1]. This popularity can be greatly attributed to the efficiency with which BitTorrent can distribute these large files.

This efficiency is partly obtained by breaking up each large file into hundreds or thousands of segments, or *pieces*, which, once downloaded by a peer, can be shared with others while the downloading continues. The sharing of pieces of the download has been shown to be very efficient [29], allowing downloads to scale well with the size of the downloading population. Another aspect of BitTorrent's efficiency comes from its resilience to peer departures, peer failures, and misbehaving peers. Many of these properties have been confirmed through both theoretical and experimental studies; however, one aspect yet to be fully explored is the topology of the network of peers formed during a download.

1.1 Piece Selection

An important consideration in controlling BitTorrent's sharing is deciding the order of pieces to download. Each peer will have to make this decision based only on the local knowledge it has of the system. An inadequate policy could lead to some pieces becoming poorly replicated, and therefore almost unavailable, while others are overly replicated, leading to starvation in areas of the system where new pieces are needed. To understand how the policy for choosing pieces in BitTorrent affects the system, it is necessary to examine the system-wide population of pieces available. This microscopic information would help to understand the dynamics and evolution of the BitTorrent swarm, and especially the effectiveness of the policy used by BitTorrent to ensure an even distribution of pieces.

We present a systematic measurement study on the distribution and evolution of the piece population in BitTorrent. Our measurement is based on real BitTorrent data gathered from both regular Internet and controlled PlanetLab swarms. The data is collected by multiple administered clients distributed in different parts of the network, which collectively offer a global view of the piece distribution.

We examine snapshots of the population of pieces in swarms, and the evolution of the piece population over several days, mostly during the early phases of a swarm's lifetime. We find that the piece distributions are generally very narrow, and progress to more narrow distributions quickly in response to changing conditions. This shows that the downloading policy of BitTorrent is effective from a piece distribution and evolution perspective, though we do find that some enhancements are possible to achieve an ideal piece distribution, especially for larger torrent swarms.

1.2 Neighbor Selection

The topology of the network of peers formed during a BitTorrent download has yet to be fully explored. In particular, the resilience to failing and misbehaving nodes suggests that the network may be *scale-free*, and the efficiency of information distribution suggests that the network may be clustered or even *small-world*. Neither of these properties has been quantitatively measured in BitTorrent, and never beyond the early stages of swarms. Since BitTorrent networks are highly dynamic, a clear understanding of the characteristics and evolution of the networks as peers arrive and depart and during their entire lifespans is critical to its performance optimization.

We describe experiments that closely examine the underlying topologies of BitTorrent swarms. These experiments capture the intricacies of forming multiple complex networks in BitTorrent, including the formation of four networks in a BitTorrent download: Connection, Interest, Unchoked, and Download. Unlike previous work which was confined to the startup stage, we look at all four networks' characteristics and dynamics throughout the entire lifespan of swarms. Our results demonstrate that the networks exhibit fundamental differences over time. This suggests that the initial stage of a BitTorrent swarm is not sufficient to predict the overall performance of the system, and in order to fully examine a BitTorrent swarm long-term measurements are needed.

We find strong evidence of scale-free characteristics in the network of peers that are unchoked by other peers. However, we find no clear evidence of persistent clustering in any of the networks of peers that we studied, which suggests an interesting venue for improving BitTorrent's performance.

We therefore present a first attempt to introduce clustering into BitTorrent. Our approach makes minimal changes to a BitTorrent tracker by adding peers to a fixed number of groups, called n-cliques. The tracker then returns mostly peers from the same group, with only a small number from other groups. Our theory indicates that this will keep the clustering high within the n-cliques, while the connections to other groups will keep the characteristic path length low. We verify this theoretical prediction using simulations and experiments.

1.3 Organization of the Work

We introduce some work related to ours in Chapter 2, and then go into some details on BitTorrent and a few network topologies in Chapter 3. The experiments that were run to measure BitTorrent's performance are described in Chapter 4. The results of the experiments on the piece selection algorithm of BitTorrent are presented in Chapter 5, while the results for the neighbor selection algorithm are in Chapter 6. A proposed enhancement to BitTorrent to make it form small-world networks, and the results of the simulations and experiments confirming its effectiveness are in Chapter 7. Finally, there follows some discussion in Chapter 8, and then the thesis concludes in Chapter 9.

Chapter 2

Related Work

Though there have been many recent research projects investigating specifically the sharing present in BitTorrent, most have focused on the macroscopic measurements of peers in the system. Izal et al. [20] gathered and analyzed long-term data from a BitTorrent tracker, but since the tracker has no knowledge of pieces or peer connections, they did not further explore these aspects. Pouwelse et al. [28] studied BitTorrent through tracker logs and also a large number of administered clients. However, they were interested mainly in measurements of the uptimes and download rates of the contacted peers. Guo et. al. [17] used a graph based multi-torrent model to study inter-torrent collaboration, but did not look at the graphs formed within a single torrent. Veciana et al. [16] created a Markov Chain model to numerically study the service capacity of a BitTorrent-like P2P system. This model was expanded on by Qiu and Srikant [29] to create a simple deterministic fluid model for the peer population of the system. They also examined the effectiveness of the file sharing in BitTorrent, using a peer-level probabilistic model that assumes peers have global knowledge.

2.1 Piece Selection

Very little previous work has explored microscopic piece-level measurements of BitTorrent. Niu and Li [26] attempted a theoretical evaluation of the *block variation* resulting from using network coding in a peer-to-peer system. Though focused on network coding, their results should be applicable by setting the size of network coding segments to 1, but they are only theoretical and do not match with our experimental results.

The closest work to ours is from Legout et al. [22], who administrated a single client and connected separately to 26 torrent swarms of differing characteristics. Their results thus reflect the piece availability only in peers their single client connected to during the experiment, which may not be representative of the entire swarm, nor does it offer global knowledge of the piece population. In contrast, our work focuses on the global piece population by examining the number of copies of each piece present in *every* peer in the swarm. We also follow the piece population over time, to see how it evolves with the swarm. Some of this work has been previously published [13].

2.2 Neighbor Selection

Recently, several authors have examined the network topologies formed by BitTorrent's neighbor selection algorithm. Urvoy-Keller and Michiardi [30] used a simulated BitTorrent overlay to look at the distance of peers from the initial seed and the matrix of peer connections. Their results were based on a homogeneous collection of peers, and were limited to the startup stage of a swarm. Al-Hamra et al. [2] expanded on those results through simulation with some experimental confirmation. They also examined the diameter of the overlay created, and the robustness of the overlay to the presence of churn and attacks. Legout et al. [23] performed an experimental evaluation with around 40 heterogeneous peers, finding interesting evidence of clustering in the network of peer unchokings.

Our results differ from these earlier results in several ways. We have focused on experimental evaluation, which captures the intricacies of the formation of multiple complex networks in BitTorrent. We use over 400 peers and explore the entire lifespans of the swarms, from the initialization stage to a steady stage. This enables us to quantitatively evaluate both time-invariant characteristics and those that evolve in different stages. Some of this work has been previously published [15].

Chapter 3

Background

We first present some basic background information on BitTorrent, followed by specific details on the areas of BitTorrent we are focused on, in section 3.1. A few network topologies, their characteristics and advantages are then described in section 3.2.

3.1 BitTorrent

We start with an overview of BitTorrent and the terminology used to describe it in section 3.1.1. BitTorrent is well-described in the literature, so more details can be found in any of the references from Chapter 2. We then discuss specific areas of BitTorrent that are relevant to our work, including the piece population in section 3.1.3, and the networks of peers formed by BitTorrent in section 3.1.6.

3.1.1 Definitions

As mentioned previously, BitTorrent breaks up each large file into hundreds or thousands of segments, or *pieces*, which, once downloaded by a peer, can be shared with others while the downloading continues. A BitTorrent *swarm* is the set of all peers currently downloading and uploading pieces to and from each other. It is made up of two types of peers, those who have the complete file (*uploaders* or *seeders*), and those who are still downloading it (*downloaders* or *leechers*). The influx or departure of peers from a BitTorrent swarm is called *churn*.

The BitTorrent system coordinates file sharing through the use of a centralized *tracker*.

Upon receiving a request from a downloading peer's client, the tracker will provide a random list of peers for the client to contact. The client will then contact each of the peers, and gather information about which pieces the peers have available for download. These *connected* peers are the immediate neighbors of the client, and all the information the client has of the system comes from knowledge of its neighbors. There is a user-configurable limit on the maximum number of neighbors a client can have, which defaults to a value of 80 in most clients. There is also a limit on the number of connections a peer can initiate, after which it will only receive new connections from other peers, which defaults to a value of 40 in most clients

Throughout the lifetime of a BitTorrent swarm, three stages are evident. The first is a *startup stage* occurring at the very beginning of the swarm, at which time only the initial seed has all the pieces of the file. Once a single copy of all pieces is uploaded to the swarm, the startup stage comes to an end and a *transient stage* begins. The transient stage is usually characterized by the rapid influx of downloaders to the swarm, which leads to a system with proportionally many more leechers than seeders. Once this influx slows, the swarm will move towards a *steady stage*, characterized by an unchanging number of seeders and leechers, so that the arrival rate of leechers must be the same as (or near to) the rate of change of leechers to seeders and the departure rate of seeds from the system. The amount of time spent in the startup stage is determined solely by the upload rate of the initial seed and the size of the file, while the time spent in the transient stage is determined by the popularity of the torrent.

3.1.2 The Rarest-First Policy

There are many policies at work in a BitTorrent client that govern how it downloads pieces. One of the most important is the *rarest-first* policy, which is responsible for choosing pieces to download with the goal of ensuring that copies of pieces are uniformly distributed throughout the system. The client constantly updates a list of the pieces each of its connected peers (neighbors) has available. Using this information, the client can determine which set of pieces it believes to be the rarest in the swarm. These rarest pieces will be selected first to download from the connected peers. Due to the limitations of the local knowledge each peer has, the pieces chosen to download may not be the rarest in the entire swarm.

A simple alternative to the rarest-first policy is to use a random piece selection policy for downloading. This policy was used by the original BitTorrent client when it was first released, but due to the superiority of the rarest-first policy it is not in use by any clients today.

3.1.3 Piece Population

The piece population is the distribution of the number of copies of each piece in the BitTorrent swarm. Each piece can have a number of copies varying from the number of seeds in the system to the total number of peers in the system (seeders and leechers). Since all seeds always have all pieces, we will restrict our discussion to the population of pieces among the downloaders in the system. Therefore, each piece in this limited view of the swarm can have a number of copies varying from 0 to the number of leechers in the system.

The population is expected to form some distribution around a mean value. This mean depends solely on the average completion of the download, which is itself determined by the arrival rate of downloaders and the transition rate of leechers into seeders. During the startup stage this mean will be low, as not all pieces are available yet in the swarm and so have 0 copies. In the transient stage the mean will be less than half the number of downloaders, as new downloaders are arriving faster than pieces can be copied in the system. In the steady stage the mean should be close to half the downloaders, as the average completion will be 50% due to new downloaders arriving in the system at the same rate as downloaders become seeders. Only at the end of the torrent's life, which we are not interested in, will the mean be larger than half the number of downloaders in the system.

Though the policy for choosing pieces to download will not have much effect on the mean of the population distribution, it will have a large effect on the width of the distribution about the mean. Ideally, this distribution width would be very small, signifying that copies are equally distributed to all pieces in the system. However, it is not hard to imagine far from optimal scenarios where this distribution could range from 0 all the way to the number of downloaders, especially considering the limited knowledge peers have of the system when it is very large. The most problems will be expected in large swarms as some peers will have chosen pieces to download that are not rare in the system, but are only rare for the local view the peer has of the system. This should create a tail in the distribution as some pieces that are not globally rare get copied to a larger number of downloaders.

The distribution width will also vary at different stages of the system, through no fault of the policy itself. For example, early on in the startup stage it is very difficult to keep a small distribution about the mean, when most of the pieces have not yet been copied in the system.

Ideally, once this stage of the swarm is complete, the distribution width should narrow very quickly as pieces are preferentially copied that were previously under-represented.

3.1.4 Block Variation

As mentioned in the previous section, we will be interested in the width of piece populations as a determination of the effectiveness of the piece selection strategy in BitTorrent. A simple choice for a width measurement would be to use the standard deviation of the distribution. However, this measurement would only be useful to compare widths measured in the same, or very similar, swarms, as the width could be biased by the mean of the distribution.

Instead of standard deviation, we will use the *block variation* introduced by Niu and Li [26] to measure the width of the piece population. If the piece distribution is represented by the random variable $\boldsymbol{I}$, which has mean μ_I and variance σ_I^2, then the block variation is defined as

$$\gamma_I^2 = \frac{\sigma_I^2}{\mu_I^2} \tag{3.1}$$

Using this unbiased measure of the width of the piece populations will allow us to compare the performance of the piece selection policy on different swarms.

3.1.5 The Incentive Mechanism

There are many policies at work in a BitTorrent client that govern how peers connect to other peers. One of the most important is the incentive mechanism, or *tit-for-tat* policy. It is responsible for choosing peers to upload to, with the goal of ensuring that peers who upload (contribute) to the system are more likely to be able to download. This game-theoretic method of encouraging sharing and fairness is built into the system to discourage free-riding peers from not uploading.

Each client notifies other peers that have pieces the client needs of the client's *interest* in downloading from the peer. A downloading client also monitors the download rates it receives from its neighboring peers. The client then allows uploads to the peers that are interested in downloading, and that it is receiving the highest download rates from. This is referred to as *unchoking* the peer. There are a limited number of unchoke slots available, so a downloading peer's neighbors will compete for its unchoke slots by uploading to it. In addition, in order to explore the connected peers and find better peers to upload to, the client will periodically choose one peer at random to unchoke, which is known as *optimistic unchoking*.

Each peer will have to make these decisions based only on the limited local knowledge it has of the system, specifically the peers it is connected to and their current uploading rates to it. A poor policy for making this decision could lead to networks that are inefficient in distributing pieces throughout the system, or are easily susceptible to disconnection due to departing or failing peers. Either of these problems would lead to a system which makes inefficient use of the the uploading bandwidth available to replicate the file.

3.1.6 Networks in BitTorrent Swarms

Given the complex relations among peers, BitTorrent actually maintains four networks within a swarm. Previous studies have focused on only one or two of the following networks, but we will investigate the properties and evolution of all four networks.

Connection Network. This is the network of neighbors that each peer maintains. These neighbors are chosen randomly by the tracker from the list of peers in the swarm. Each peer creates connections to the peers returned by the tracker (up to its limit on initiating connections) and also makes return connections to other peers that connect to it (up to its maximum neighbor limit). All neighbor connections are bi-directional, so the Connection Network is undirected.

Interest Network. This network represents the interest that peers have in other peers. Each peer maintains a list of the pieces stored by its neighboring peers. A peer is interested in any neighboring peer that has a piece it does not have, and so this network is a subset of the Connection network. Since interest can be uni-directional, the Interest Network is directed.

Unchoked Network. This network is formed by the incentive mechanism present in BitTorrent. Each uploading peer assigns its limited number of unchoke slots to certain neighboring peers in an effort to maximize the downloads that it receives from them. Only peers that are interested in receiving an upload are unchoked, so this network's transpose is a subset of the Interest network. Since unchoking can be uni-directional, the Unchoked Network is directed.

Download Network. This network is formed by the peers that are downloading from other peers. Since a peer has to be unchoked before it can download, this network is a subset of the Unchoked network's transpose. Since downloading can be uni-directional, the

Download Network is directed.

We emphasize that the Connection and Unchoked networks are the most important of these four networks. The Connection network forms the neighbor set for all of the peers in the system, and is a superset of the other three networks. The Unchoked network is necessary for the uploading and downloading of data from other peers, and so it is very important for the scalability and efficiency of BitTorrent.

3.2 Network Topologies

The neighbor selection algorithm in BitTorrent will lead us to examine two types of network topologies: scale-free networks, described in section 3.2.1; and small-world networks, described in section 3.2.2.

3.2.1 Scale-Free Networks

The existence of scale-free graphs in many real-world networks was first introduced by Albert and Barabási [7]. They found that in many real networks, such as the network of actor collaboration in Hollywood, the U.S. airline system, or the world wide web, the distribution of node degrees follows a power-law. These seemingly randomly-formed networks thus exhibit a complex topology not accounted for in random graph theory. More specifically, independent of the system and the identity of its participants, the probability $P(k)$ that a node in the network is connected to k other nodes decays as a power law, given by

$$P(k) \sim k^{-\gamma} \tag{3.2}$$

in which the power γ is usually found to be between 2 and 3 in various real networks [6]. This results in a large number of nodes having a small node degree and therefore very few neighbors, but a very small number of nodes having a large node degree and therefore becoming *hubs* in the system.

To determine if the node degrees in the network exhibit a power law distribution and to measure its exponent, we will plot the degree of each node against the rank of the node by degree on a log-log scale [4], as shown in Figure 3.1. The slope of a linear fit then yields the power law exponent, and an R^2 goodness of fit value can also be generated to indicate the accuracy of the fit, and therefore verify the presence of a power-law distribution.

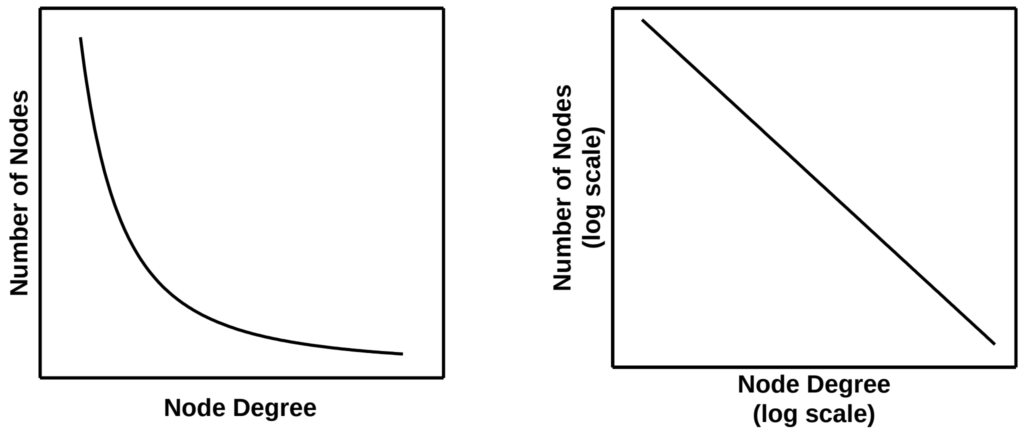

Figure 3.1: A power-law distribution of node degrees.

The scale-free nature of many real-world networks was found to be a consequence of two mechanisms that were present in the construction of the network. The first is that the network was formed over time, and is continually evolving as new nodes join the network. The second is that nodes have distinguishing characteristics, and new nodes will attach preferentially to existing nodes that have desirable characteristics. Both of these conditions also exist in many peer-to-peer networks [33, 34], including BitTorrent. A BitTorrent network is constantly evolving, as new nodes join the system over time, complete their download, remain to upload to others, and then eventually leave the system. Peers in a BitTorrent system will also have desirable characteristics, such as a large uploading bandwidth, or possession of many or all of the pieces in the system. Nodes in the BitTorrent system can then attach to these nodes and choose them for unchoking or downloading from. However, the presence of scale-free characteristics in BitTorrent swarms has not been previously confirmed.

The desirability of scale-free graph characteristics comes from these networks' tolerance to random node failure [3]. Due to the presence of hubs in the network, a random loss of a large percentage of the network (as much as 80%) will not result in degraded network connectivity. In a peer-to-peer system such as BitTorrent, where random node failures or departures (churn) are quite common, this resilience is necessary to maintain efficient sharing of the file by all nodes in the system. This is especially important for the Unchoked network generated by the incentive mechanism, as node failures should not result in a decreased usage of the system's total uploading or downloading bandwidth. The tolerance of scale-free graphs to node failures has also been shown to extend to misbehaving nodes in the

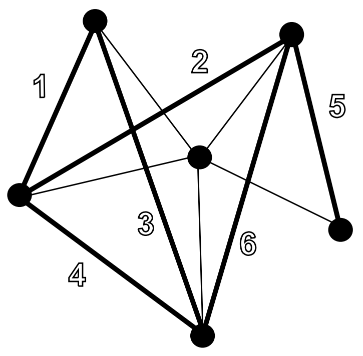

Figure 3.2: The neighborhood of a single node with a clustering coefficient of 0.6. The 6 of the 10 possible edges that exist between neighbors of the node are in bold and numbered.

system. This is also desirable in BitTorrent to avoid performance problems that could occur due to free-riders in the system.

3.2.2 Small-World Networks

The concept of a small-world phenomenon was first introduced by Milgram [25] to refer to the principle that people are linked to all others by short chains of acquaintances (popularly known as *six degrees of separation*). This formulation was used by Watts and Strogatz to describe networks that are neither completely random, nor completely regular, but possess characteristics of both [31, 32]. They introduce a measure of one of these characteristics, the cliquishness of a typical neighborhood, as the *clustering coefficient* of the graph. They define a small-world graph as one in which the clustering coefficient is still large, as in regular graphs, but the measure of the average distance between nodes, the *characteristic path length*, is small as in random graphs.

Given a graph $G = (V, E)$, the clustering coefficient C_i of a node $i \in V$ is the proportion of all the possible edges between neighbors of the node that actually exist in the graph. A sample graph showing a single node's neighbors and its clustering coefficient is shown in Figure 3.2. For a node i of degree k_i, the maximum possible number of undirected edges between neighbors of the node is $k_i(k_i-1)/2$; for directed edges, this is doubled. This can be visualized as the number of triangles that exist in the graph which include the node as one of the vertexes of the triangle. The clustering coefficient of the graph $C(G)$ is then the average of the clustering coefficients of all nodes. Watts and Strogatz show that many real-world networks exhibit small-world behavior [32]. They calculate the clustering coefficient for some networks to be: 1 for cliques, 0.79 for the network of collaborating actors in Hollywood, a maximum of 0.75 for circulant graphs, 0.28 for the neural network of *Caenorhabditis elegans*, and almost 0 for random networks.

Small-world networks are desirable features to have in a communication system, and especially in peer-to-peer file sharing systems, as they are expected to be efficient at delivering messages to all nodes in the system. Latora and Marchiori [21] measured the efficiency of the information exchange in small-world networks. They find that the small-world networks are both globally and locally efficient at exchanging information over the network. This was also examined by Comellas et. al. [11, 12], who looked at broadcasting and the spreading of epidemics in small-world communication networks. They find that the networks are very efficient at both broadcasting and the spreading of viruses, both of which are similar to the distribution of a file in a file-sharing system such as BitTorrent.

Many existing peer-to-peer systems (e.g., Gnutella [24], Freenet [18,35], and DHT-based systems [19]) are known to be small-world. It is natural to expect that BitTorrent swarms would exhibit small-world characteristics, particularly since clustering has been previously observed in the early stages of swarms [23].

Chapter 4

Experimental Design

All experiments were run on PlanetLab using a modified BitTorrent client as described in section 4.1. Experiments that measure the piece population of swarms are described in section 4.2, and experiments that measure the topology of BitTorrent swarms are described in section 4.3.

4.1 Running BitTornado on PlanetLab

We gathered all experimental data using a modified BitTornado program [8], which is a typical and widely-used BitTorrent client. We modified both the default behavior (through options), and the inner workings of the program. These changes allowed us to log the pieces downloaded by the client, and the connections made between clients for the four types of networks described in section 3.1.6. We also collected data on the pieces available in other unmodified clients we connect to, through the sharing of available piece information that is built into BitTorrent's protocol. Except for these changes, we have made no modification to the normal operations of the BitTorrent client and protocol.

The modified BitTornado client was used to collect data from up to 450 nodes of the PlanetLab research network testbed [27]. We used the PLDeploy program created by Mic Bowman to add our modified client to multiple PlanetLab nodes, control its operation, and collect the results. In some experiments, only a small number (10 to 20) of PlanetLab nodes were used to collect information from other uncontrolled peers in a real Internet swarm. These nodes were configured to connect to as many peers as quickly as possible, using multiple requests to the tracker for peers. In other experiments, all available nodes (up to

Table 4.1: The distribution of peer characteristics used in the simulated BitTorrent swarms.

Nodes	**Duration**	**Upload**	**Download**	**Unchoke Slots**[a]
45 %	44 hours	6 kB/s	6 kB/s	2
25 %	20 hours	12 kB/s	24 kB/s	3
15 %[b]	12 hours	25 kB/s	75 kB/s	4
10 %	6 hours	50 kB/s	150 kB/s	5
5 %[c]	4 hours	100 kB/s	500 kB/s	6

[a] This is the number of unchoke slots available for uploading (out-degree). Downloading (in-degree) is not limited.
[b] In one experiment all the nodes were in this class.
[c] The original seed for the experiment was in this class of peers.

450) were used in a *simulated* BitTorrent swarm, in which all the peers were controlled by us and ran our modified client. After the experiments, the data was synchronized for time by examining the times that pairs of peers logged a bidirectional connection.

For the simulated swarms, we created a test file consisting of 780 MB of random data (a typical size for a BitTorrent download) and assigned one node to be the original seed. The connection speeds that we used are shown in Table 4.1, and are typical of those available from Internet Service Providers. In one experiment all peers used the same connection speeds, in which case they were taken from the middle group. In other experiments, the percentage of nodes chosen from each group (shown in Table 4.1) for each connection speed was determined from our previous measurements of real internet swarms. The number of unchoke slots was varied according to recommended values for different connection speeds [5]. We varied the limit on the maximum number of neighbors each peer can have in some experiments, but usually left it set at 80 which is the default value in many BitTorrent clients.

The clients were scheduled to join randomly over the first 4 hours of the experiment, download the file to completion, and then seed it to other downloaders. In some experiments, to realize a larger total number of peers than is possible with only 400 nodes, the clients were scheduled to seed for a random period, then leave, delete the file and rejoin as a new peer to restart the download. The average durations spent in the system are shown in Table 4.1 (except for the original seed, which stayed indefinitely). These experiments were run for over 100 hours, which was enough time for all of the peers to become seeds, leave

Table 4.2: The torrents used for the piece population experiments.

Torrent Name	Pieces	Size (MB)	Leechers	Clients[a]
KNOPPIX[b]	4125	4325	169	10
FreeBSD[b]	5699	1494	34	10
mandriva[b]	2803	735	89	9
openSUSE[b]	14805	3881	398	9
feisty[c]	1387	727	65-120	20
openSUSE-2[c]	14977	3926	100-150	18
PlanetLab-1[c]	1497	784	0-340	340
PlanetLab-2[c]	1497	784	0-390	390

[a] Number of administered clients used to connect to the swarm. All peers are administrated clients in PlanetLab experiment
[b] Snapshots of population taken
[c] Evolution of population monitored

the system, and rejoin multiple times.

4.2 Piece Population

In some experiments, we examined the piece populations of real Internet swarms, as described in section 4.2.1. In other experiments, we examined simulated swarms which we ran on PlanetLab, as described in section 4.2.2. The characteristics of all the torrents used in the piece population experiments, both real and simulated, are shown in Table 4.2.

4.2.1 Real Internet Swarms

For these swarms, we used multiple administrated clients to record the piece population in a real BitTorrent swarm. We began with snapshots of the piece population by monitoring the number of peers contacted by our clients to determine when most of the swarm had been contacted, at which point the experiment was terminated. These experiments took less than one hour to contact over 90% of the peers in a swarm[1], which is enough to get a clear view of the global piece population.

We also studied a swarm's evolving piece population over time. This is useful to study

[1] We were able to determine the actual total number of peers from the tracker's website.

how the piece population varies at different stages, especially when the measurement starts in an initial transient stage. The method is the same as the snapshot, except that the experiment is run for a much longer time.

We chose 6 torrents to monitor from well-known Linux distributions that use BitTorrent to distribute their files, which are shown in Table 4.2. They are all freely available online, and also have tracker information available on the web. Their file sizes are representative of the file sharing being done by BitTorrent.

4.2.2 Simulated PlanetLab Swarms

In the two *simulated*[2] PlanetLab swarms, all the clients are controlled by us and use the modified BitTornado program, which enables us to have an even closer look at the piece distribution and evolution. We created a sample torrent file that resembles real ones, shown in Table 4.2, and ran a tracker for the purpose of the experiments.

In the PlanetLab-1 swarm, the downloading and uploading bandwidths of the clients are all set to 75 and 25 KB/s respectively, and we restricted each client's maximum number of neighbors to 40 to further enhance the locality effects of the piece population. All other parameters were left at the default values for the client.

In the PlanetLab-2 swarm, the distribution of bandwidths shown in Table 4.1 was used, and the clients were made to leave and rejoin continuously. We also added alternating periods of heavy and light churn to the swarm by grouping the arrival and departure of peers. This swarm is more typical of real BitTorrent downloads, and the long period of time will allow us to see what happens to the evolving piece population, and what effects churn has on it.

4.3 Network Topologies

We conducted several experiments to examine the network topologies of peers formed by BitTorrent's neighbor selection algorithm. All the experiments were run on all available PlanetLab nodes, and all used the distribution of peers shown in 4.1. The peers were made to leave and rejoin continuously, and all experiments proceeded for upwards of 100 hours.

[2]We say simulated though these swarms are real in the sense that they consist of real BitTorrent clients connecting to each other from different parts of the network. They are only simulated in the sense that we control all of them.

In our main experiment, the limit on the maximum number of neighbors was kept at the default value of 80, no additional churn was added to the system, and a normal BitTorrent tracker was used.

To investigate the impact of some key factors, in particular the connectivity, churn and our proposed enhancement, we conducted additional experiments. In one experiment, we removed the limit on the number of neighbors that each peer could have. In another experiment, we introduced alternating periods of high and low churn. In our final experiment we used a modified tracker to try and introduce small-world properties to the resulting swarm. We found that most of the results are similar to the main experiment, so we will only highlight the differences.

All of the figures were created by measuring the characteristics of the networks at regular intervals during the experiment. Approximately 360 networks were generated for each figure from the 100 hours of data. We determined this to be frequent enough to capture all of the details of the evolving networks.

The Connection, Interest, and Unchoked graphs are constructed using actual connections among peers that have a defined start and end time. However, the Download graph is more difficult to construct because downloading from a peer occurs almost instantaneously for the small piece size. To overcome this difficulty, we constructed it by considering all peers that have downloaded from each other since the last measurement to be connected in this network.

It is worth noting that some of the directed Interest, Unchoked, and Download networks are not connected, due to the presence of seeds. Seeds have the entire file, so they are not interested in other peers, are not unchoked by other peers, and do not download from other peers. This can cause problems with calculations that depend on the network being connected. For calculations based on these networks we used the largest strongly connected component, which is equivalent to removing the seeds from the network as well as any rare poorly connected nodes.

Chapter 5

Piece Population

We now present and analyze our measurement results for the piece populations of the torrent swarms shown in Table 4.2. The results of some snapshots we took of the piece population in some real Internet swarms is shown in section 5.1. The evolution of 2 real internet swarms over long periods of time is shown in section 5.2. Finally, we ran several simulated swarms on PlanetLab, the results of which is in section 5.3.

In order to make visual comparisons between different swarms, some normalization of the data is needed. For all data, we normalize the number of copies (x-axis) by the total number of downloaders, so that it varies from 0 to 1. If the populations are all from the same swarm, the population size data (y-axis) will be normalized by the number of pieces, so it will also vary from 0 to 1. However, this normalization does not make sense when comparing different swarms' populations, as it leads to a much smaller population when the number of pieces is larger. Therefore, to facilitate the comparison of multiple swarms, they will be normalized so that the area under their population distribution is 1. Since we are only interested in the width of these distributions, this normalization should have no effect.

5.1 Snapshots

Figure 5.1 shows the snapshots of the piece populations for various real torrent swarms. All four appear to be normally distributed with mean values slightly less than half the downloaders, indicating that they are in the transient stage. The least normally distributed populations are Knoppix and openSUSE, which correspond to their being the largest swarms. This larger swarm size results in peers having a limited local view of which pieces are rarest,

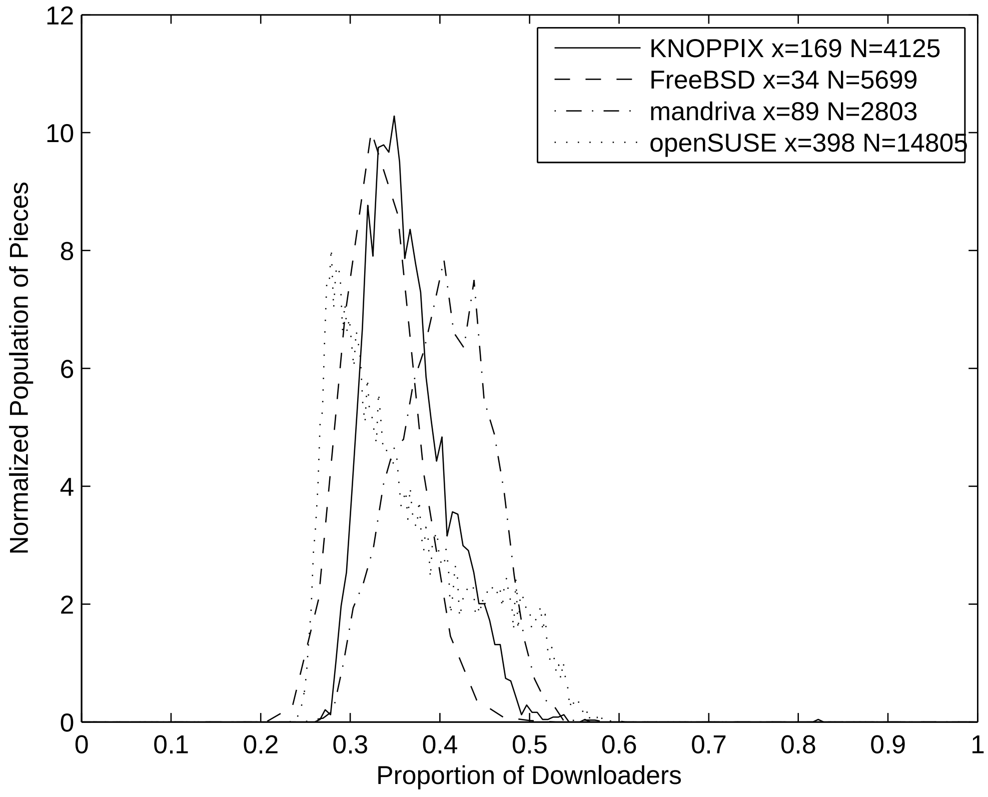

Figure 5.1: The piece population snapshots of four real Internet swarms (x = number of downloaders, N = number of pieces).

Table 5.1: The block variation for the snapshot measurements of real internet swarms.

Torrent Name	**Pieces**	**Leechers**	**Block Variation**
KNOPPIX	4125	169	5.4×10^{-4}
FreeBSD	5699	34	1.0×10^{-4}
mandriva	2803	89	4.4×10^{-4}
openSUSE	14805	398	2.0×10^{-3}

which leads to a distortion of the normal curve towards some pieces having extra copies (the tails evident in Figure 5.1). The other two populations are small enough that a peer's local view is nearly complete, resulting in a near perfect normal distribution.

Table 5.1 confirms our visual analysis of the width of the distributions. The largest swarm (openSUSE) has a block variation that is an order of magnitude larger than the others. The smallest block variation is for the FreeBSD swarm, which also has the fewest number of peers.

5.2 Evolution

To further understand the dynamics of piece population in the different stages of the swarm, we have also monitored swarms throughout their lifetime. It is worth noting that such experiments can be difficult to conduct for real Internet swarms because, in general, we do not know the exact start time of a swarm unless it is launched by ourselves. Swarms launched by ourselves, however, are not necessarily representative and the measurement results can be biased. Through constant online tracking, we did find several swarms that we began monitoring very early, and we now show two of them.

5.2.1 The feisty Swarm

Figure 5.2 shows the evolution of the feisty swarm over a period of 17 hours. The monitoring began soon after the torrent was launched, and though the number of leechers has already peaked, a peak is clear in the number of seeders near the middle of the experiment. This leads to the conclusion that this swarm was in a transition from the transient stage to the steady stage as the experiment progressed. However, there was a large influx of peers near the 12 hour mark, probably due to a news posting.

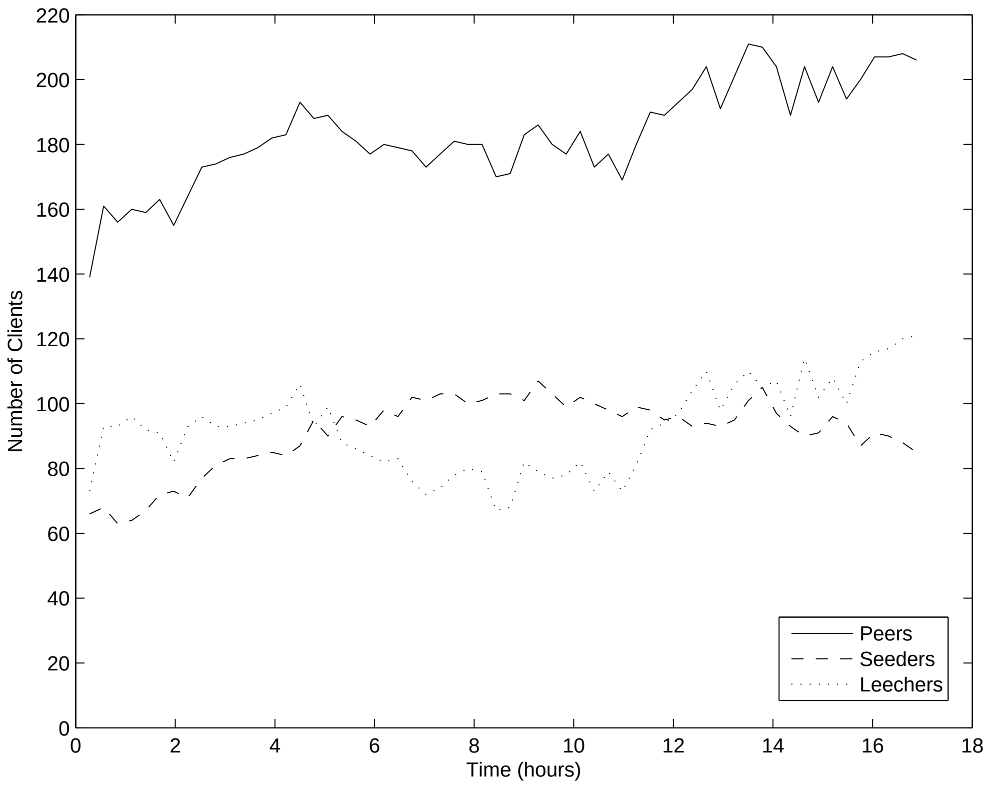

Figure 5.2: The number of peers in the feisty swarm.

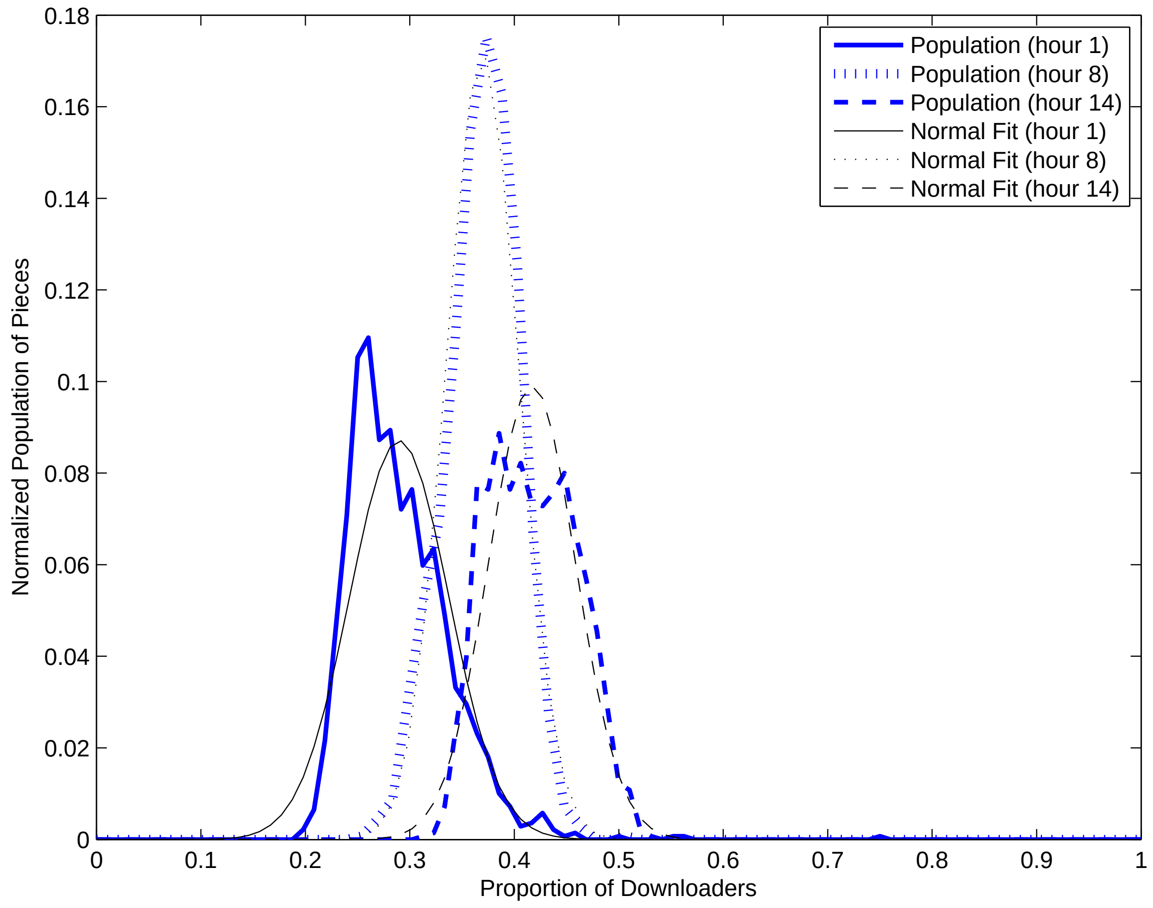

Figure 5.3: Selected piece populations from the feisty swarm.

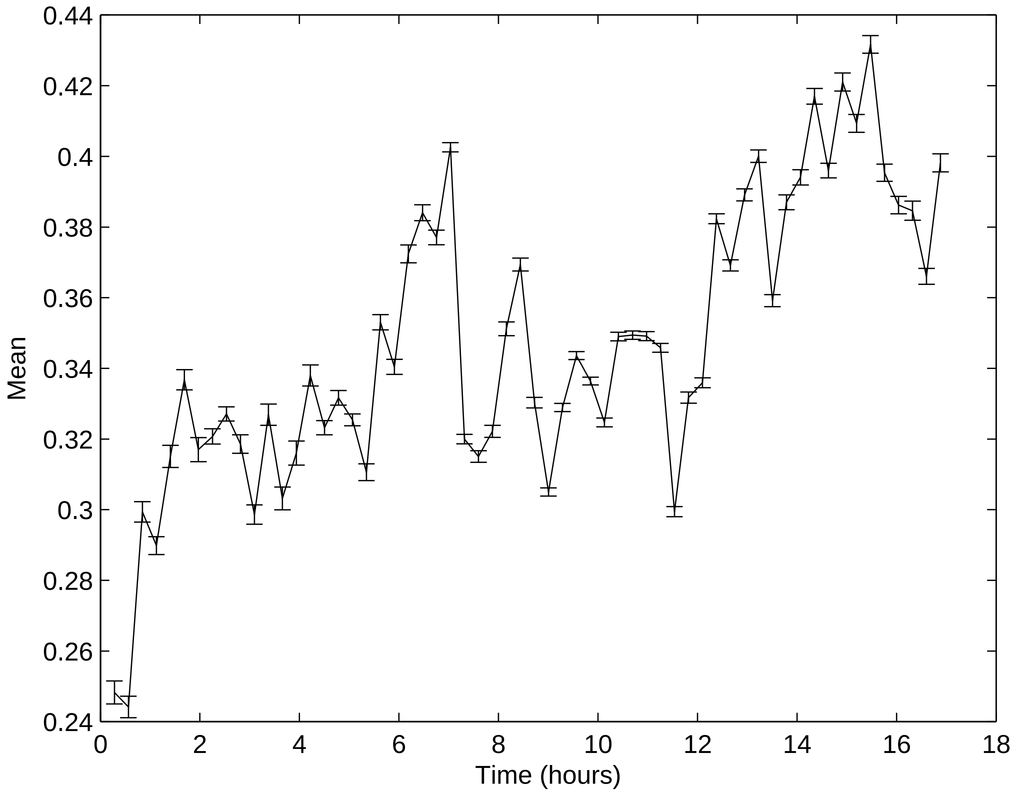

Figure 5.4: The mean of the piece population in the feisty swarm (error bars are 95% confidence intervals).

Figure 5.3 shows three representative plots of the piece population in the feisty swarm, as well as fitted normal distributions. The piece population is seen to be progressing towards a more normal distribution early in the experiment, although it does increase in width towards the end of the experiment, as noted below by the block variation.

The mean is progressing towards a value of 0.5. This can be clearly seen in Figure 5.4, though somewhat noisy. Figure 5.5 shows the progression of the block variation towards a more narrow distribution, reaching a minimum block variation of 2×10^{-4}. However, the width does increase by an order of magnitude near the end of the experiment after the large influx of peers occurs.

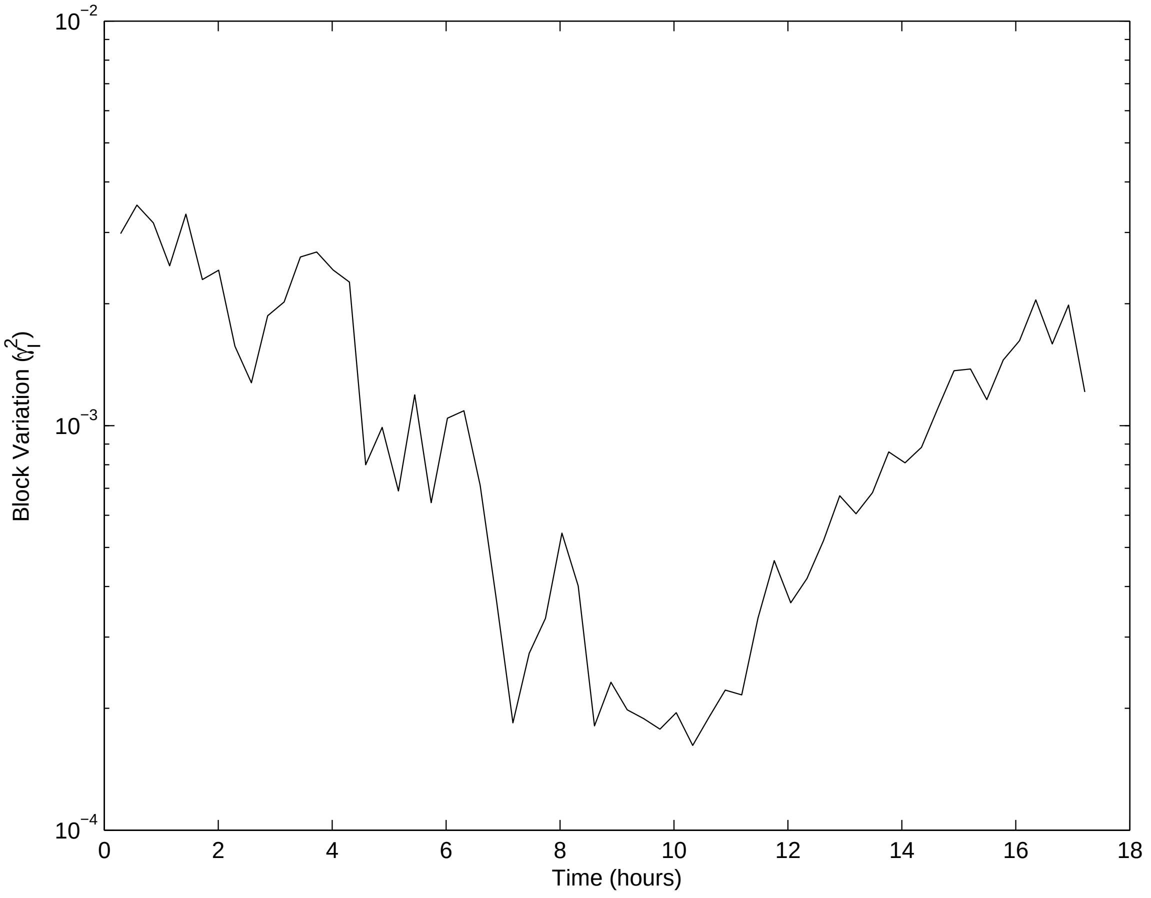

Figure 5.5: The block variation of the piece population in the feisty swarm.

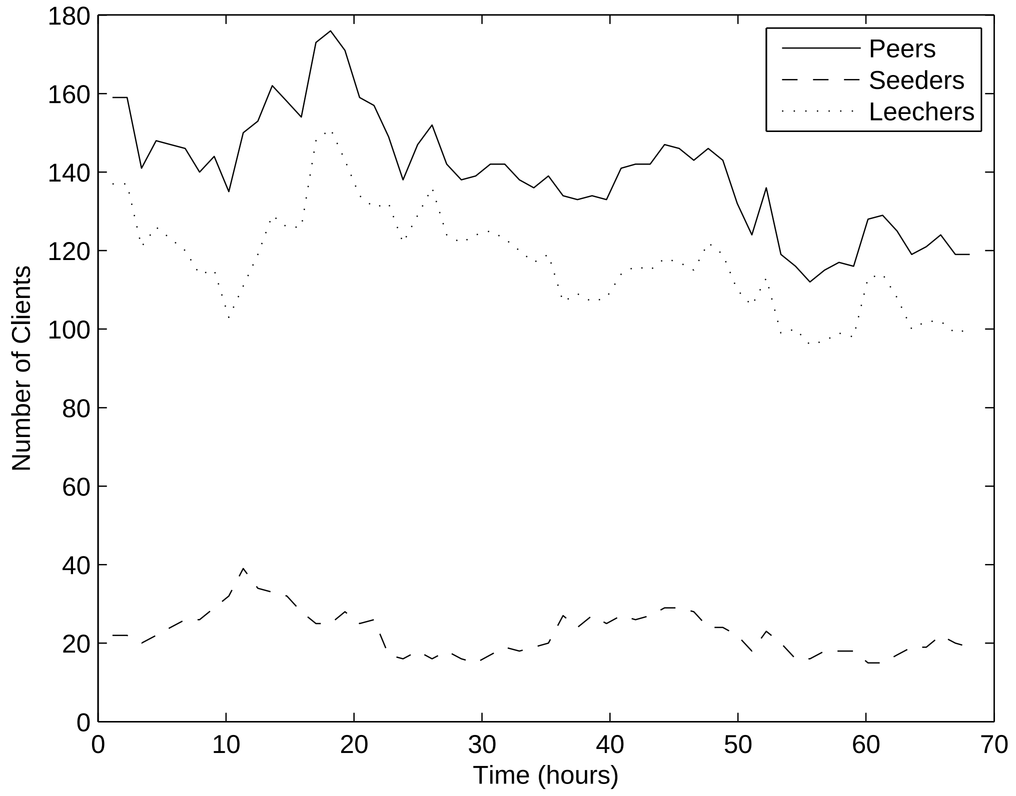

Figure 5.6: The number of peers in the openSUSE-2 swarm.

5.2.2 The openSUSE-2 Swarm

Figure 5.6 shows the evolution of the openSUSE-2 swarm over the period of 70 hours while it was being monitored. The monitoring began very soon after the torrent was launched, as seen by the large number of leechers and few number of seeders. Due to the size of the download, not much change is seen in the swarm during the experiment, though the number of leechers decreases without an increase in the number of seeders, indicating that seeders are leaving the system at the same rate as peers are becoming seeders while few new peers are joining the system. This system is considered to still be in the transient stage.

Figure 5.7 shows three representative plots of the piece population in the openSUSE-2 swarm, as well as fitted normal distributions. The piece population is very clearly seen to be progressing towards a more normally distributed shape, as the population shown for

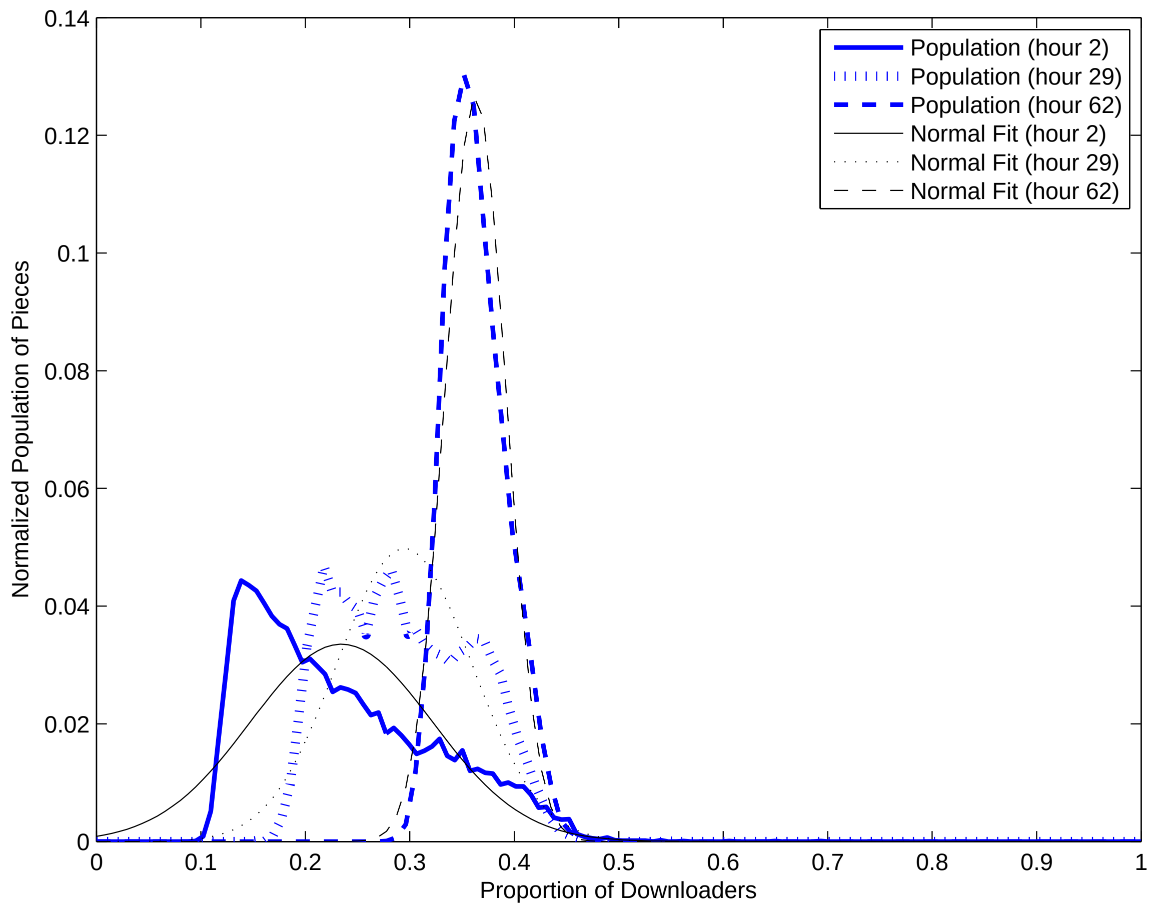

Figure 5.7: Selected piece populations from the openSUSE-2 swarm.

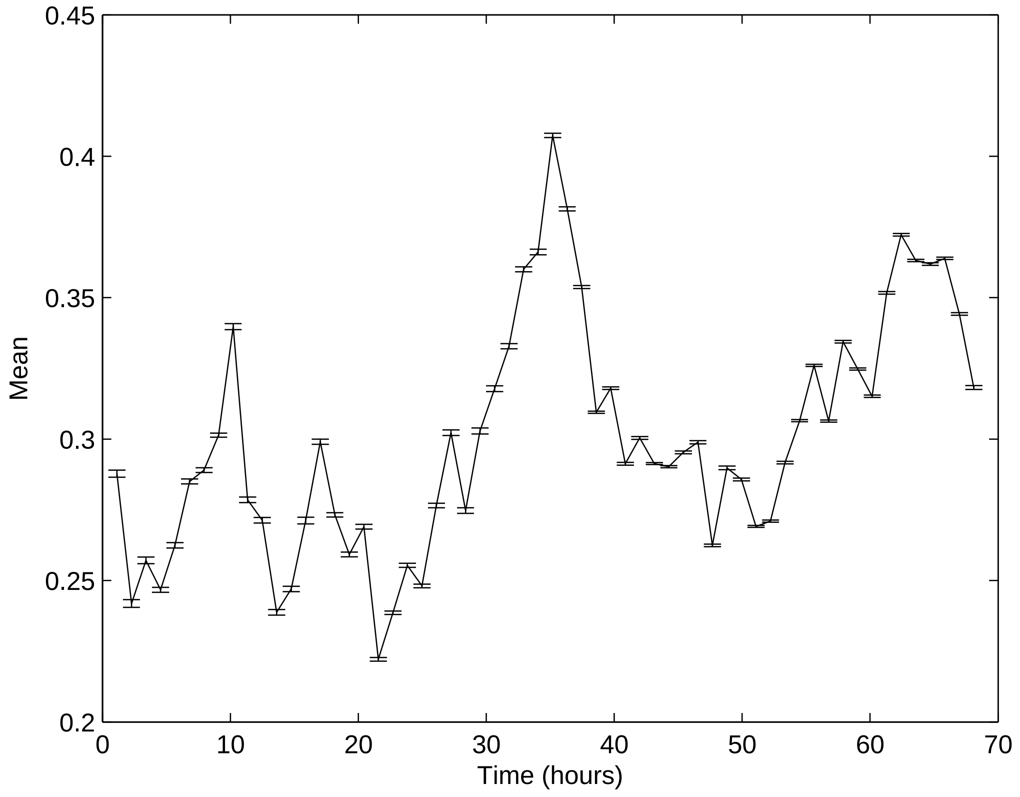

Figure 5.8: The mean of the piece population in the openSUSE-2 swarm (error bars are 95% confidence intervals).

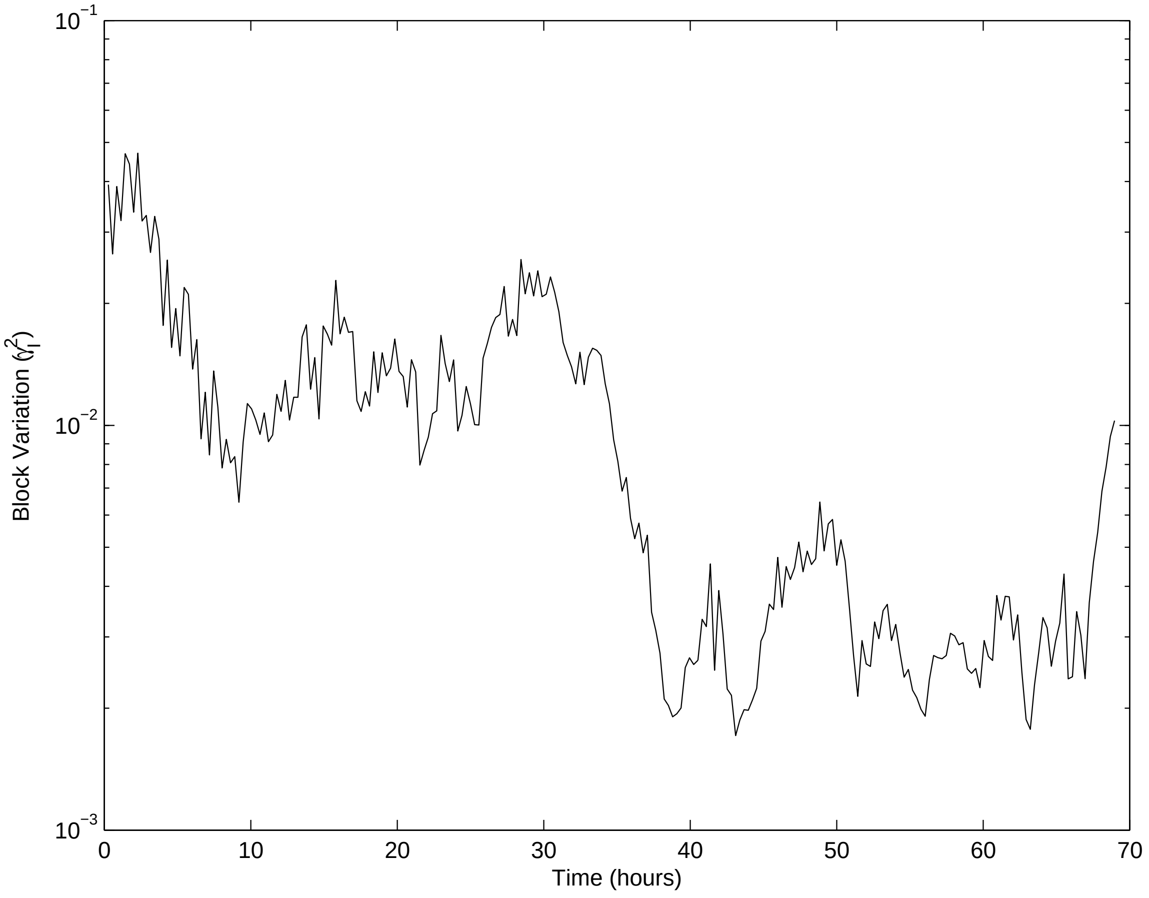

Figure 5.9: The block variation of the piece population in the openSUSE-2 swarm.

hour 2 has a very large tail. The mean is also shown in Figure 5.8 to be clearly progressing towards higher values. Figure 5.9 also shows the clear progression of the block variation towards a more narrow distribution of pieces. However, in this swarm the block variation does not reach any smaller than 10^{-3}. This is believed to be due to this swarm being in an earlier stage than the feisty swarm was, and the slightly larger number of leechers in the swarm. The size of the download is also much larger, which may have an effect on the block variation, especially at an early stage of the swarm.

5.3 Simulated PlanetLab Swarms

The controlled PlanetLab environment enables us to closely investigate the piece population of a swarm in any period throughout its lifetime, and to introduce interesting factors that it may be hard to find in real Internet swarms. We ran several swarms on PlanetLab to investigate these possibilities. The first is a short-lived swarm, for which the results are in section 5.3.1. We then simulated a much longer-term swarm that also included periods of increased churn, shown in section 5.3.2. That section also includes some results from a third swarm that used a random piece selection policy for comparison.

5.3.1 The PlanetLab-1 Swarm

Figure 5.10 shows the evolution of the PlanetLab-1 swarm over the period of 12 hours that we simulated it. This system is in the startup stage (as seen by the single seed that is available) through most of the experiment, transitioning to the transient stage after approximately 9 hours. The system then moves quickly to an end stage not seen in the other real swarms, as no new clients join the system but many are completing their download, and many are leaving the system.

Figure 5.11 shows three representative plots of the piece population in the PlanetLab-1 swarm, as well as fitted normal distributions. The normal distribution does not match the first two plots at all, as most pieces suffer from a low replication rate, while peaks higher in the population show some pieces have a much higher replication rate. This is due to the limited upload bandwidth of the original seed, and the time it takes for a single copy of the file to be present in the network (approximately 9 hours). However, the third population at 10 hours shows that the swarm takes very little time to become very narrowly distributed after the first full copy of all pieces are present in the network.

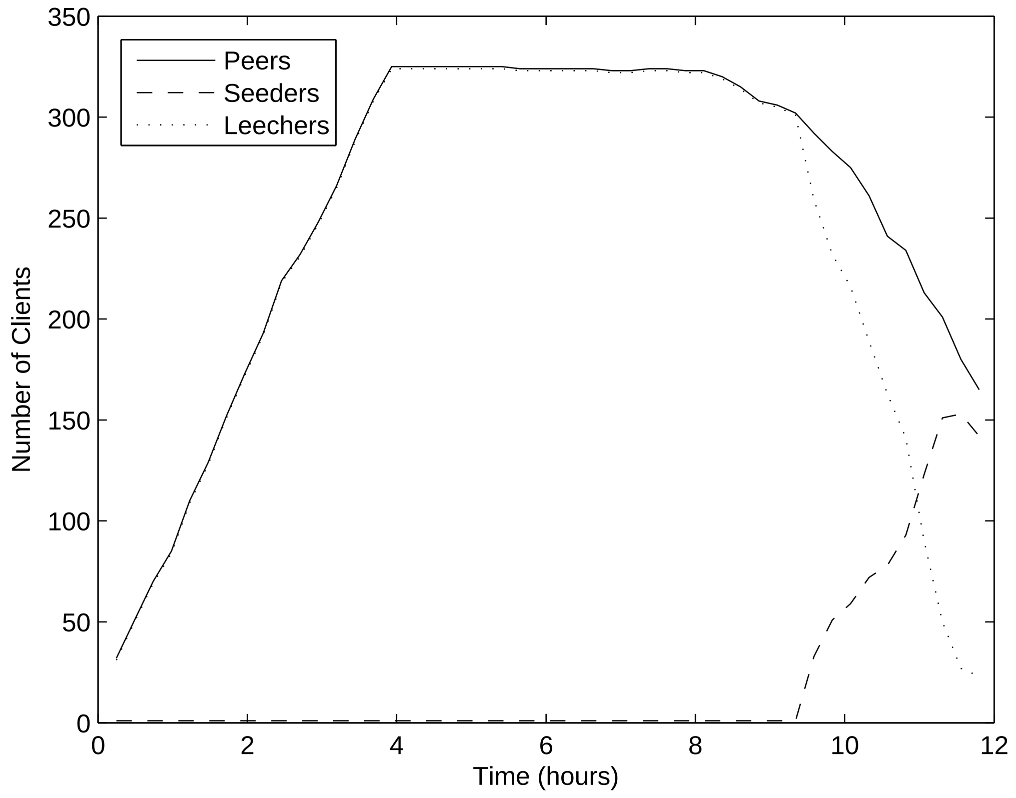

Figure 5.10: The number of peers in the PlanetLab-1 swarm.

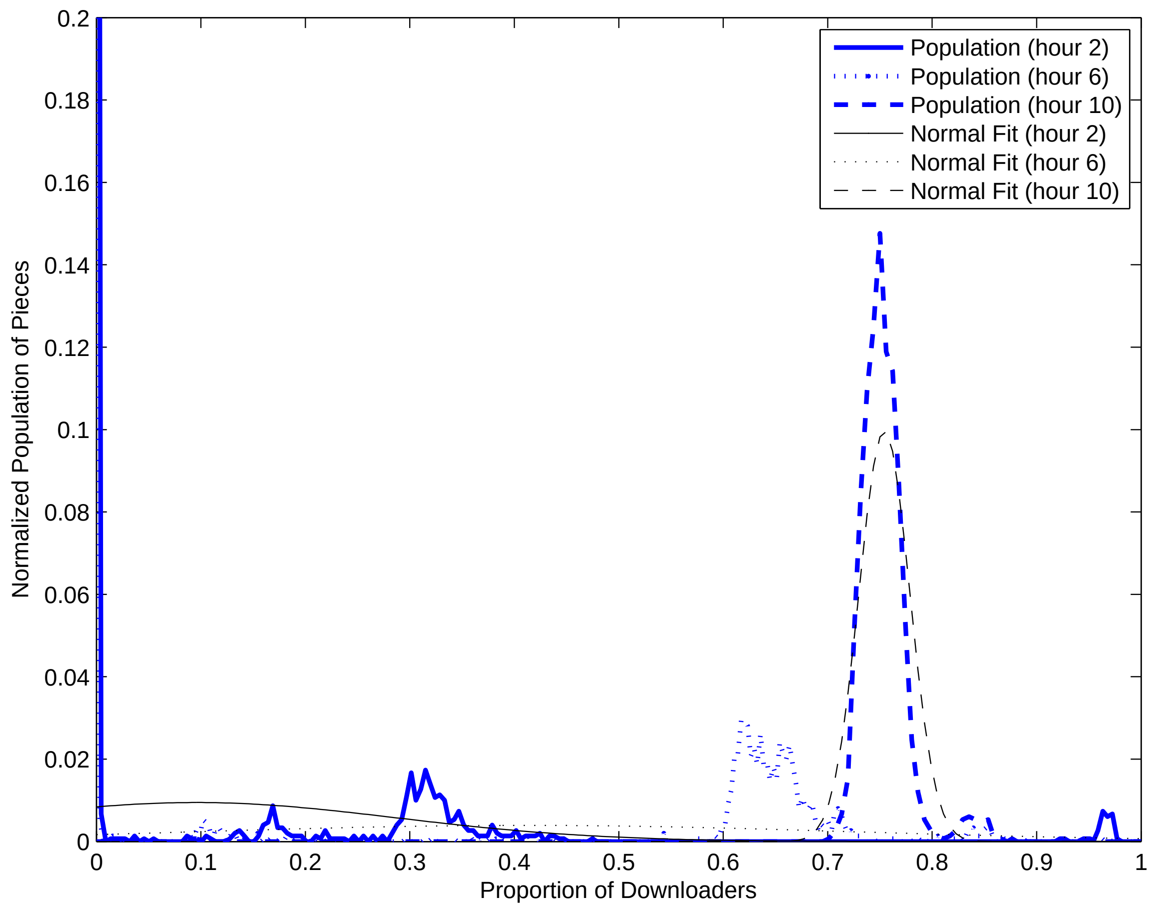

Figure 5.11: Selected piece populations from the PlanetLab-1 swarm.

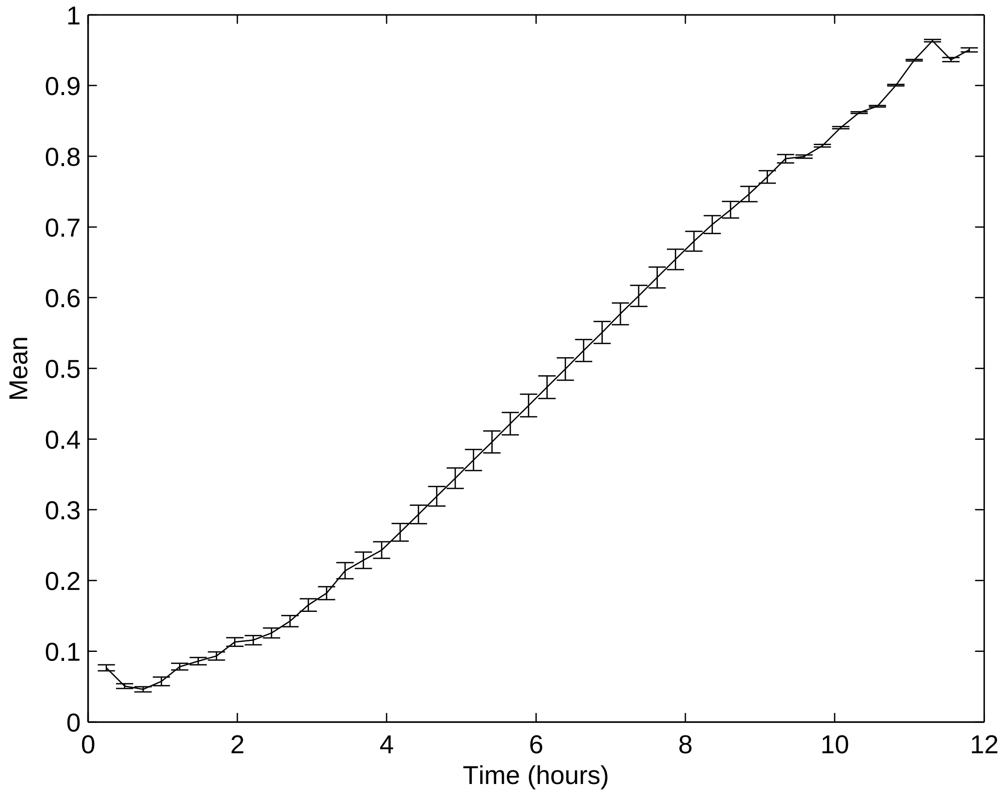

Figure 5.12: The mean of the piece population in the PlanetLab-1 swarm (error bars are 95% confidence intervals).

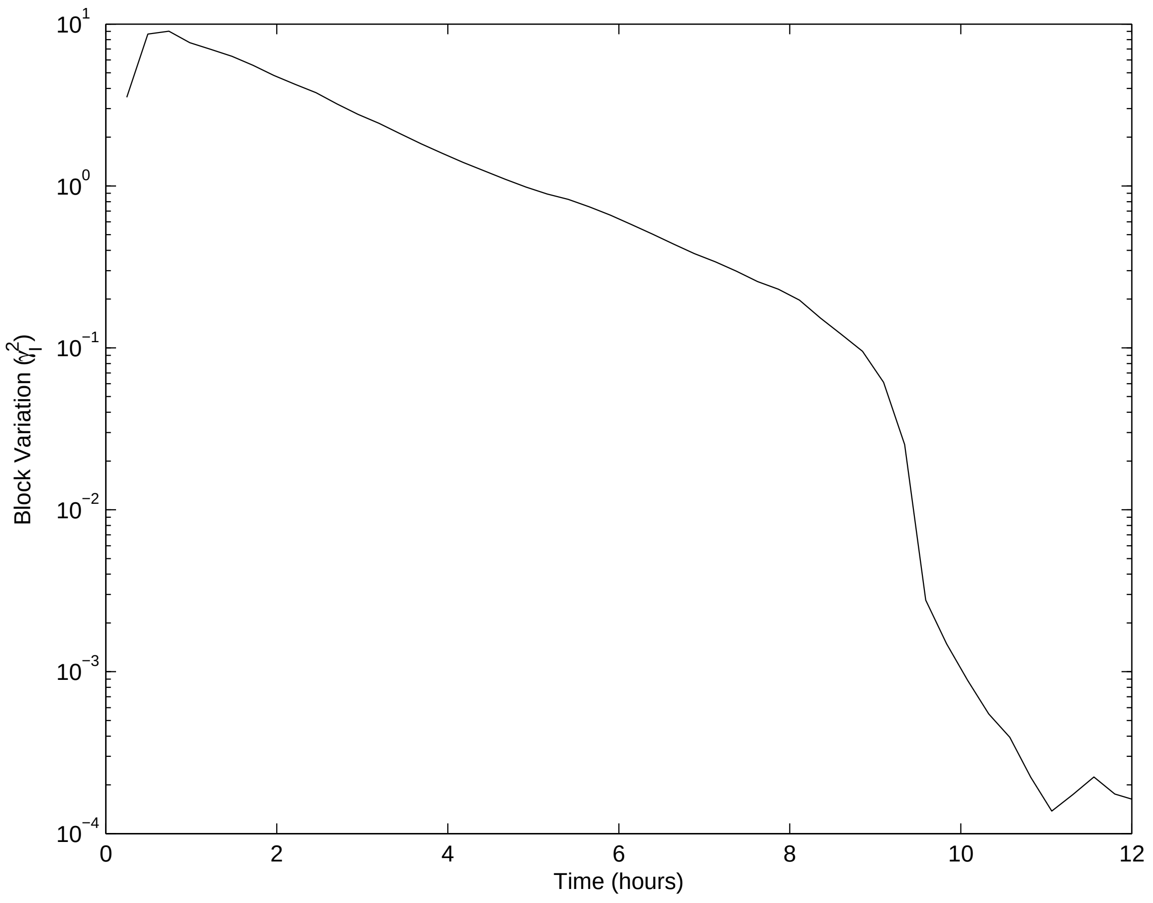

Figure 5.13: The block variation of the piece population in the PlanetLab-1 swarm.

The mean in this experiment, shown in Figure 5.12, progresses linearly towards a higher value, finishing very close to 1. Figure 5.13 shows the rapid progression of the block variation towards a very narrow distribution of pieces around the mean value once the first copy of the file is present in the network (at 9 hours). The large variation in the piece population in the first 6 hours is expected, as the piece population goes from an initial stage of very narrow (all pieces have no copies), to a split population (some pieces have no copies), and finally to a normal distribution of narrow size soon after all pieces enter the system. The block variation again reaches a value close to 10^{-4} by the end of the experiment.

5.3.2 The PlanetLab-2 Swarm

In this experiment, to simulate a more real-world scenario, we have used the distribution of upload and download speeds shown in Table 4.1. Peers are also made to restart the download multiple times by leaving and reconnecting without the file. To evaluate the impact of churn on the piece population, we varied the amount of churn at certain points in the system by grouping the peer departures and arrivals together.

Figure 5.14 shows the resulting population of seeders and leechers in the system over the period of 120 hours that we simulated it. Although the total number of peers does not change, increased periods of churn occur when the number of seeds decreases rapidly, for example from 20 to 25 hours, 40 to 50 hours, and 60 to 70 hours. Since the number of peers is limited, the periods between these increased churn periods exhibit a state of decreased churn as compared with the previous experiment.

The progression of the mean in this experiment is shown in Figure 5.15. The changes in the amounts of churn are obviously having an effect on the mean, also causing it to oscillate. Figure 5.16 shows the rapid progression of the block variation towards a very narrow distribution of pieces once the initial stage is complete. After that, the block variation again reaches a low value of 10^{-4}.

The periods of churn clearly have an effect on the block variation, causing it to periodically increase by an order of magnitude to 10^{-3}. However, the block variation does not always increase during the periods of churn. This may be due to unseen details in the types of peers that are departing or arriving. For example, the increased block variation may only occur when peers with very fast download or upload speeds arrive or depart. Nevertheless, the piece selection policy is very effective at reducing the block variation back down to 10^{-4} once the periods of churn are over.

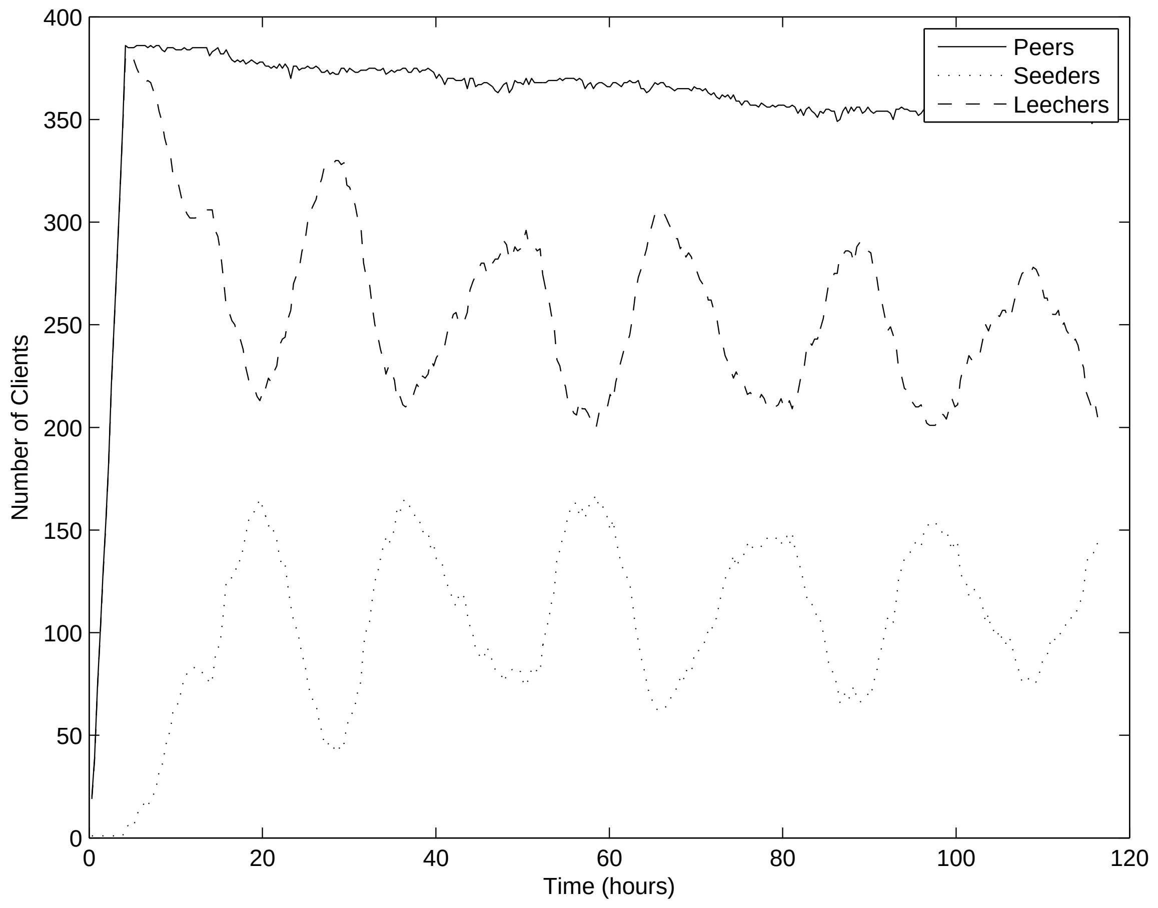

Figure 5.14: The number of peers in the PlanetLab-2 swarm.

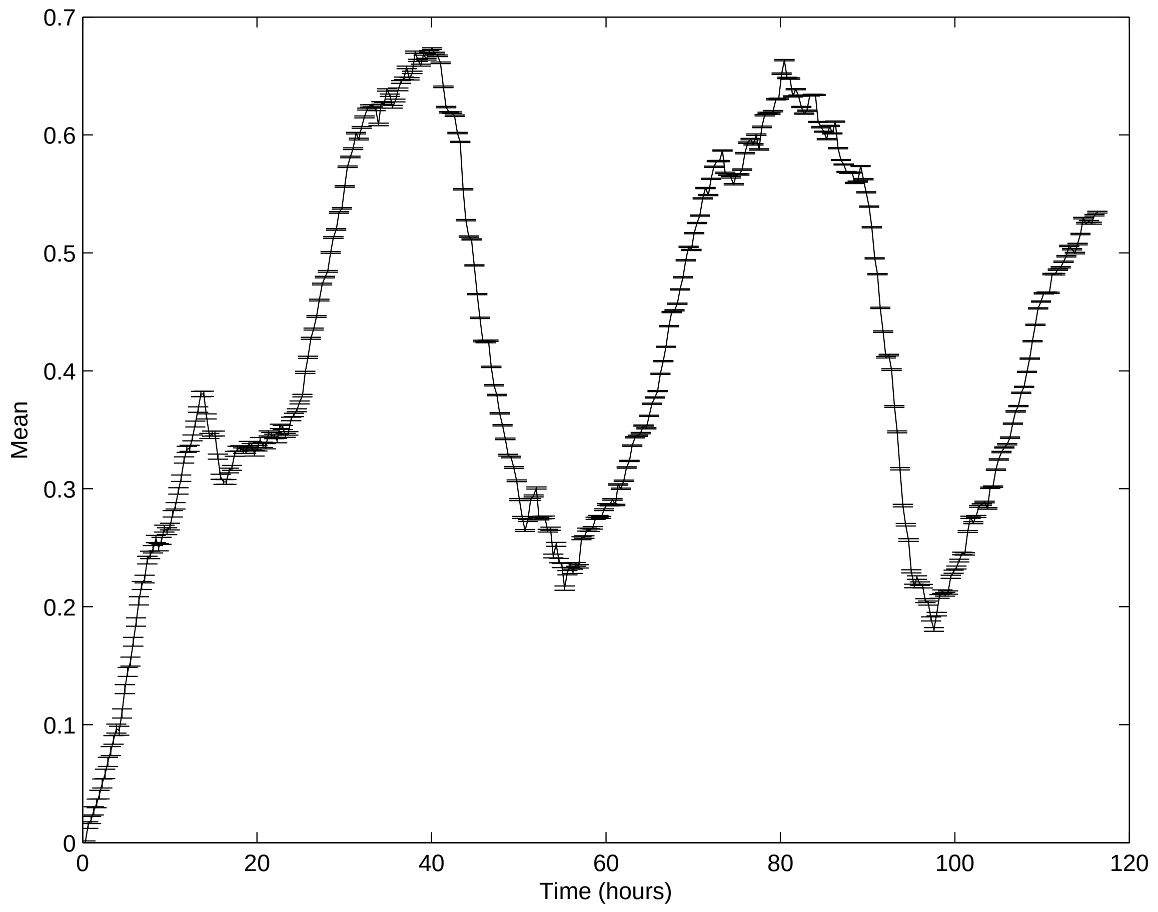

Figure 5.15: The mean of the piece population in the PlanetLab-2 swarm (error bars are 95% confidence intervals).

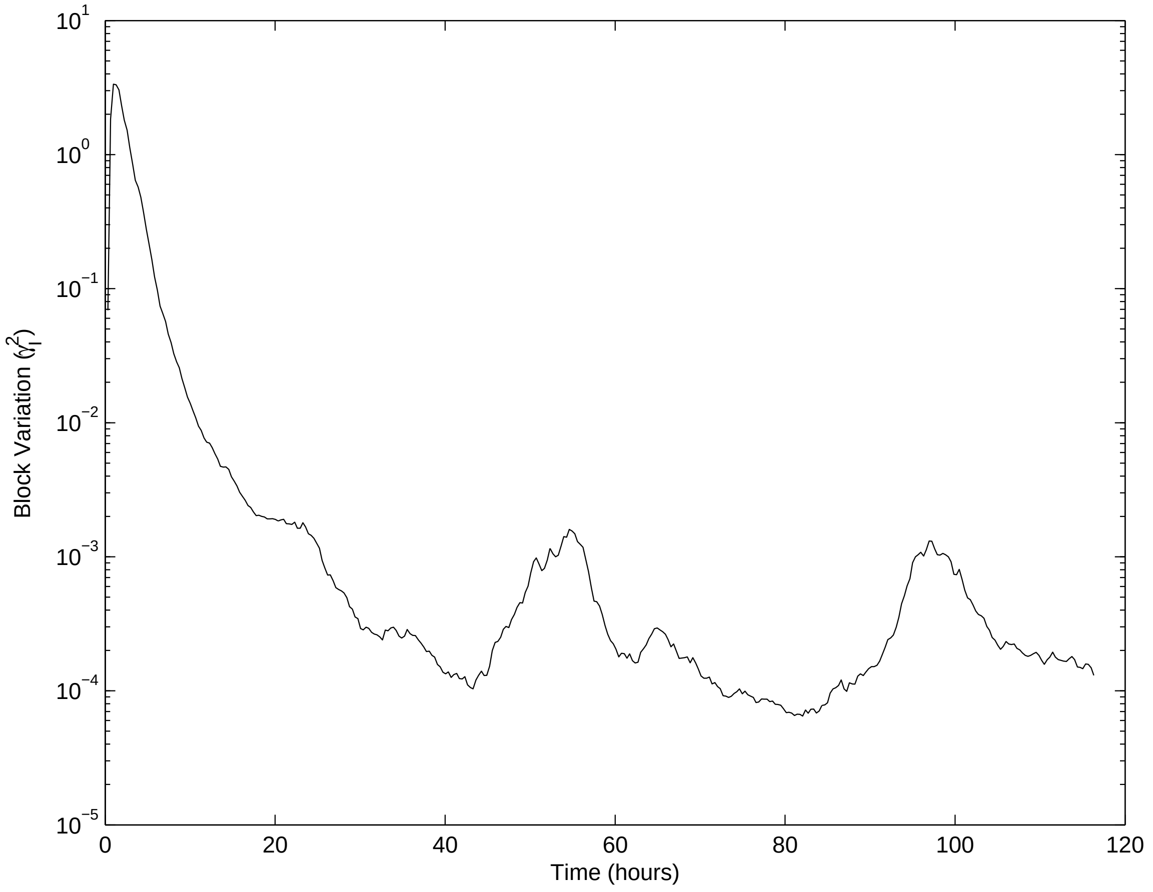

Figure 5.16: The block variation of the piece population in the PlanetLab-2 swarm.

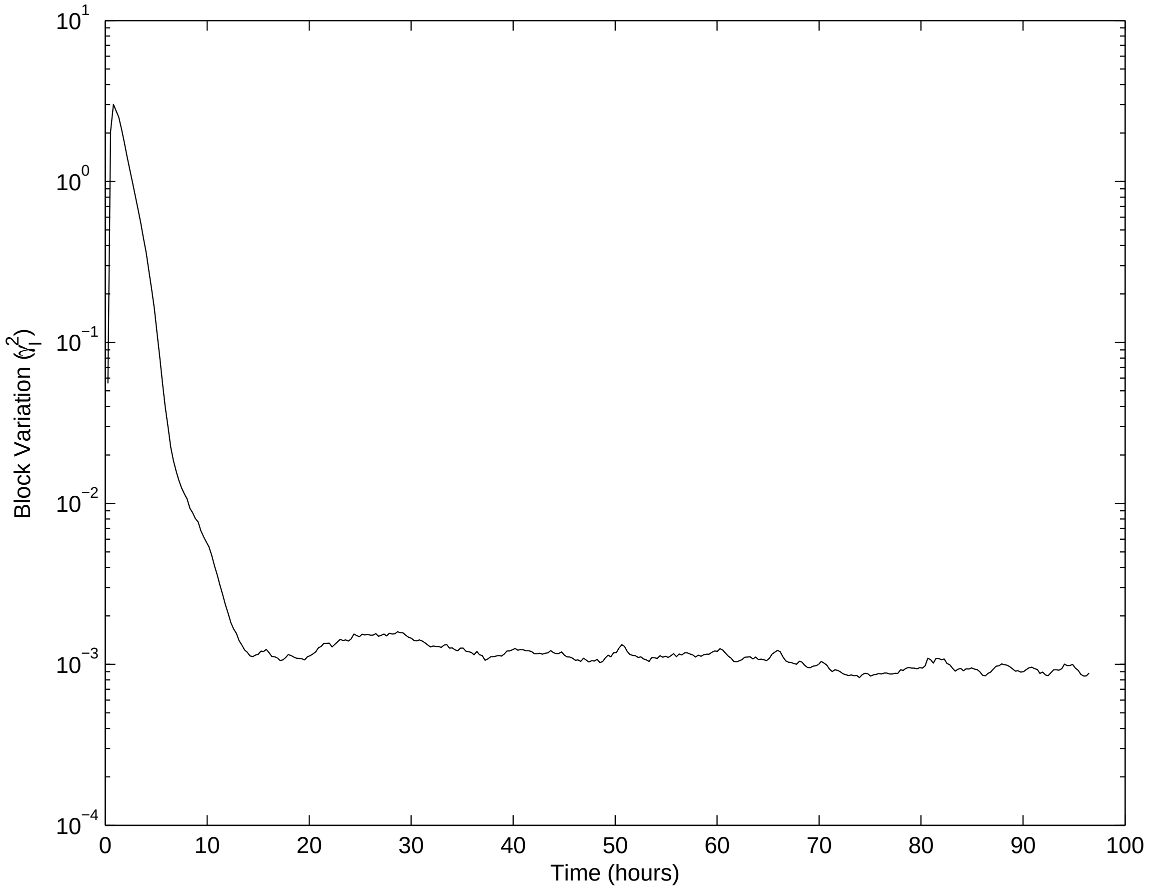

Figure 5.17: The block variation of the piece population using a random piece selection algorithm.

For further comparison of the effectiveness of the rarest-first piece selection policy, we performed an additional experiment in which we replaced it with a random piece selection policy. In this experiment, we also removed the increased periods of churn, to allow the policy to work as best it could to reduce the block variation. This experiment is otherwise unmodified from the PlanetLab-2 experiment.

Figure 5.17 shows the block variation that occurs when the rarest-first piece selection algorithm is replaced with a random one. Although the algorithm had plenty of time to reduce the block variation, the lowest achieved variation is only 10^{-3}, which is an order of magnitude larger than the smallest block variation achieved by the rarest-first piece policy, even in the presence of increased churn.

Chapter 6

Network Topologies

We now turn our attention to the measurement of BitTorrent's neighbor selection algorithms. Because peers in BitTorrent do not share information on which other peers they are connected to, the only way to gather this information is to control all the peers ourselves. Therefore, all the experiments in this chapter were run on simulated PlanetLab swarms. All the experiments also use the distribution of peers shown in Table 4.1, and have nodes leave and rejoin to lengthen the experiment's measurement time.

We will first look at the characteristics of the networks formed in our experiment in section 6.1. The connectivity matrix of the peers in the experiment is then presented and analyzed in section 6.2. Finally, we will present the differing results of another experiment with increased churn in section 6.3.

6.1 Characteristics of Network Topologies

Figure 6.1 shows the distribution of seeds and leechers throughout the first 45 hours of the 100 hours of the experiment. The remaining 55 hours are not shown as the system has reached a steady stage after which there is little change in the results.

We can identify roughly three regions in Figure 6.1. The system starts out in a startup stage with a single seed, and then peers join randomly. After the first 4 hours, the total number of peers in the system is approximately 430. Some of the peers are already beginning to be converted into seeds before 4 hours as they joined early with the fastest download rate and so have already completed their downloads. The system then enters a transient stage, from 5 hours to about 25 hours, during which the numbers of seeders and leechers

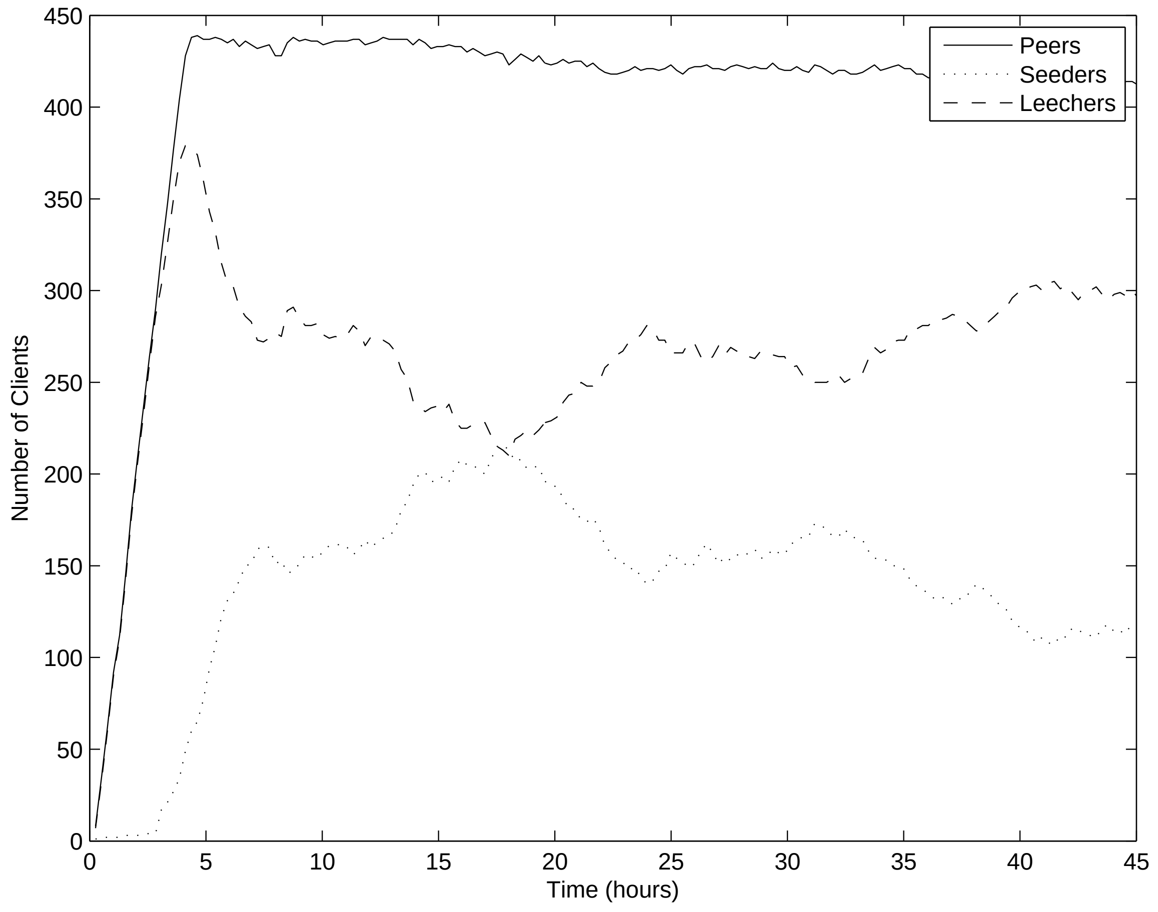

Figure 6.1: The population of peers in the system during the first 45 hours of the experiments.

in the system are still varying. After 25 hours, the numbers of seeders and leechers change very slowly and are generally quite steady. From 45 hours to 100 hours (not shown in Figure 6.1), the numbers of seeders and leechers change by less than 10%, probably due only to the randomness built into the experiment.

6.1.1 Scale-Free Characteristics

As described in section 3.2.1, we will determine if a network is scale-free by doing a linear fit to the node degree plot on a log-log scale. A sample node degree distribution and fit is shown in Figure 6.2.

Figure 6.3 shows the R^2 goodness of fit values for the in-degree of the four networks

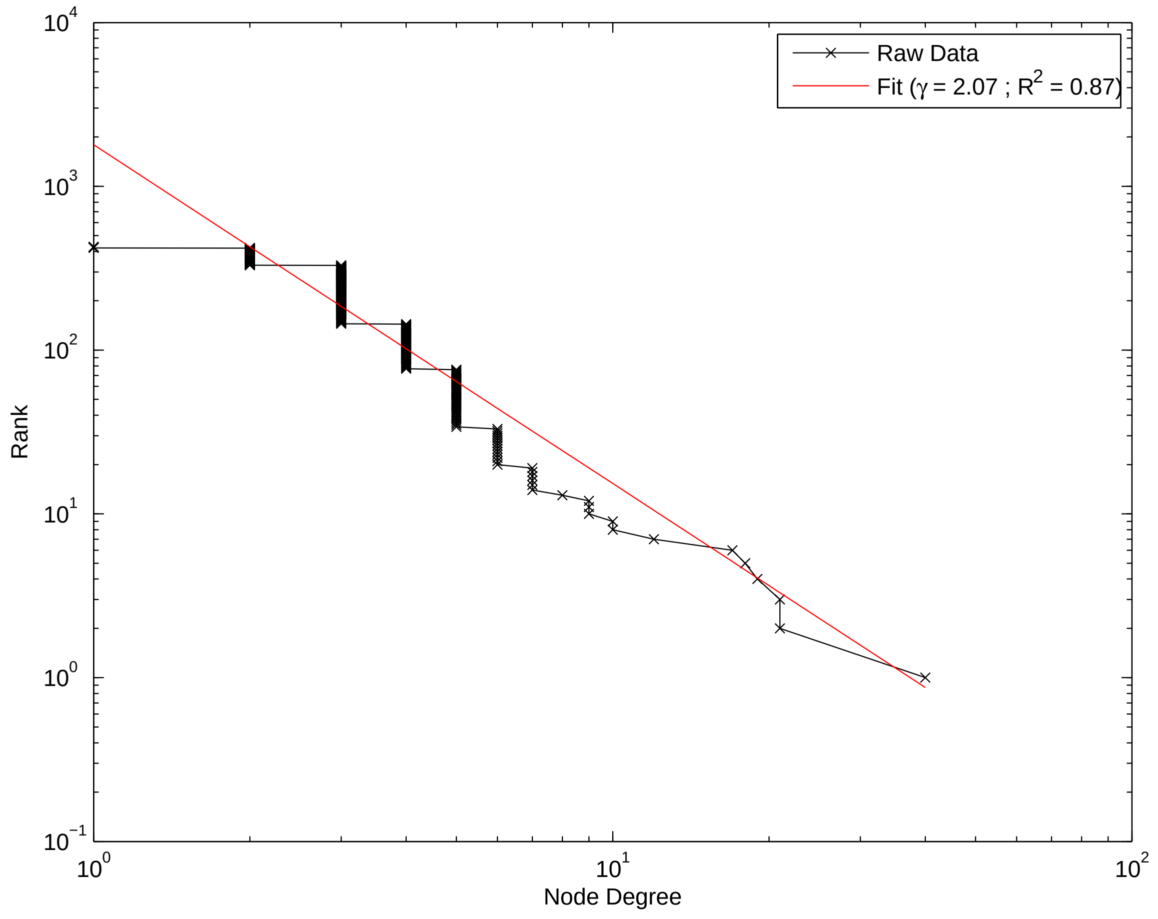

Figure 6.2: The node in-degree distribution for the Unchoked network at hour 19 of the experiments, and the resulting fit to it.

throughout the experiment. The only network of the four that exhibited this power law behavior was the Unchoked network, which had an R^2 goodness of fit value of approximately 0.9 over most of the experiment (except during the startup stage). This is high enough to indicate a good fit, while the other networks had goodness of fit values less than 0.7.

Figure 6.4 shows the power law exponent found from the fitting of the in-degree of the nodes in the Unchoked network. The power law exponent can be seen to vary quite a lot during the initial stage. However, once all the peers have joined the system the power law exponent quickly reaches its final value, and remains very steady at just over 2 through most of the transient stage and all of the steady stage.

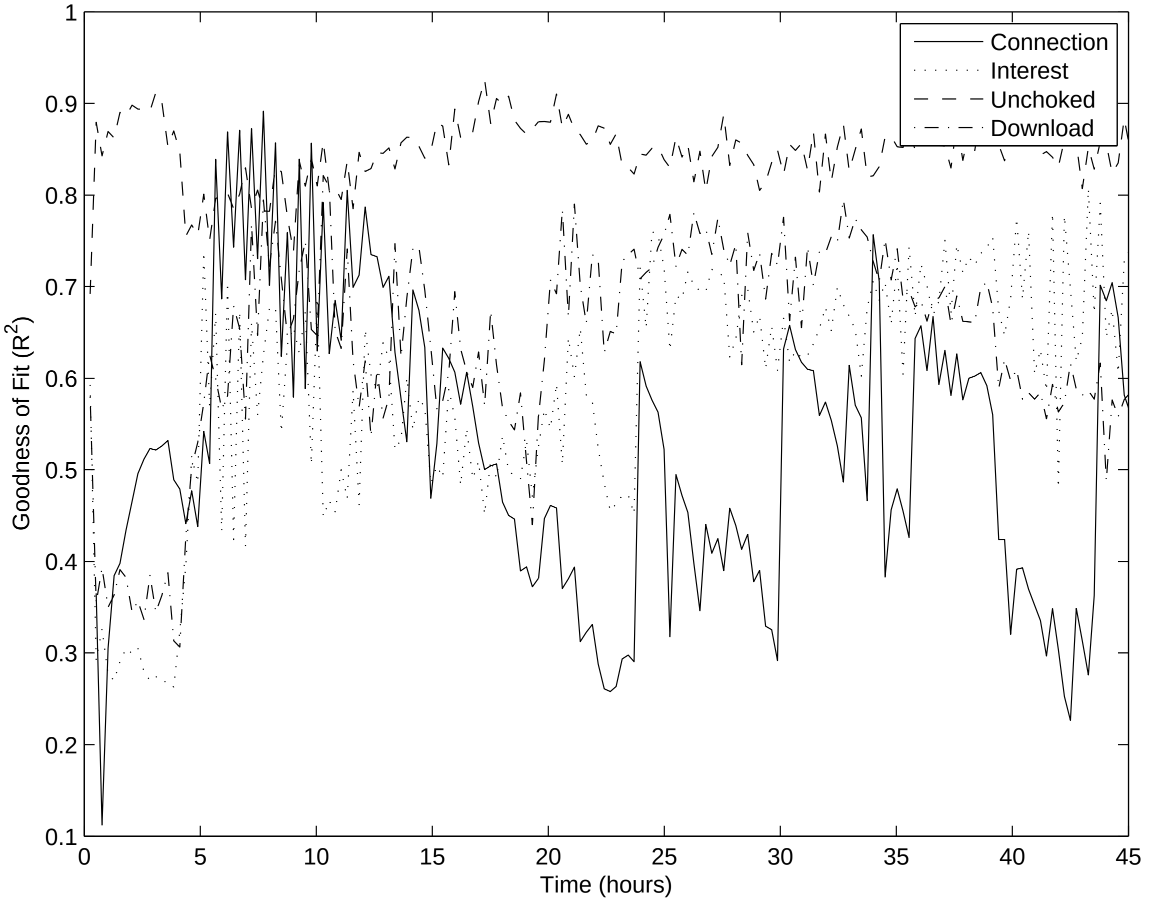

Figure 6.3: The goodness of fit (R^2) parameter measured during the experiments.

6.1.2 Small-World Characteristics

Figure 6.5 shows the characteristic path lengths of the four networks in the experiment. Note that, for the directed Interest, Unchoked, and Download graphs, the path lengths were calculated on the graphs after they were reduced to their largest strongly connected component to avoid the disconnected nature of BitTorrent graphs. The characteristic path length increases rapidly during the startup stage, though all but the Unchoked network slow their increase even before the startup stage is complete. The Unchoked graph reaches its final value early in the transient stage, after which none of the networks vary much at all.

The characteristic path lengths for the Connection, Interest, and Download networks are short, due mostly to the density of the graph (430 nodes with an average degree of

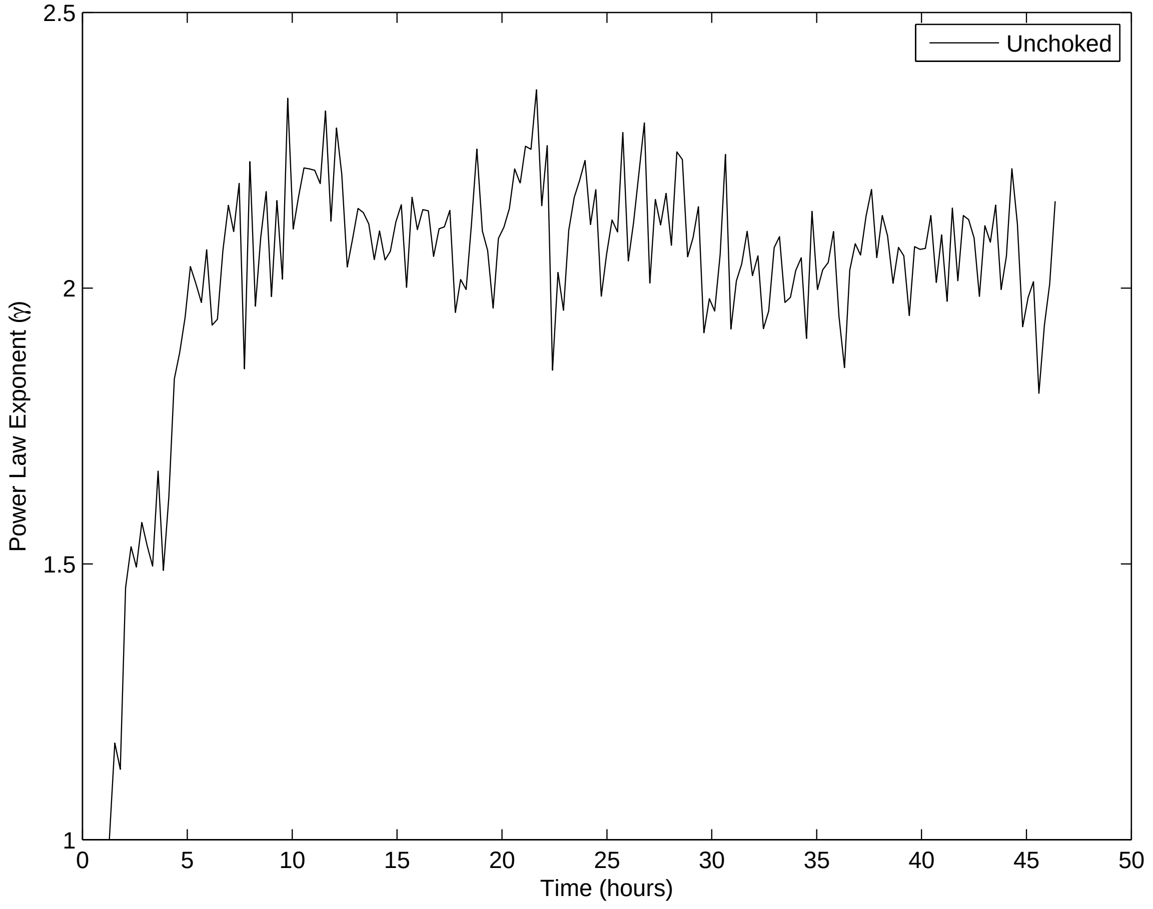

Figure 6.4: The exponent found by fitting a power law to the Unchoked graph's node degree during the experiments.

65). The Unchoked graph's characteristic path length is larger due to the reduced average degree (about 4) of nodes in this graph. Also shown are the characteristic path lengths of a randomly constructed graph [9] with the same number of nodes and edges, and with similar limits on the node degree. The random graph results are almost not visible, as the Connection, Interest, and Download graphs have nearly the same characteristic path lengths as their random graph counterparts. The only exception is the Unchoked graph which is about 10% larger, probably due to the scale-free nature of this graph which causes it to vary slightly from being truly random.

Figure 6.6 shows the clustering coefficients of the four networks in the experiment. Although not shown in the figure, the coefficient starts at 1 (since it is a clique), and then

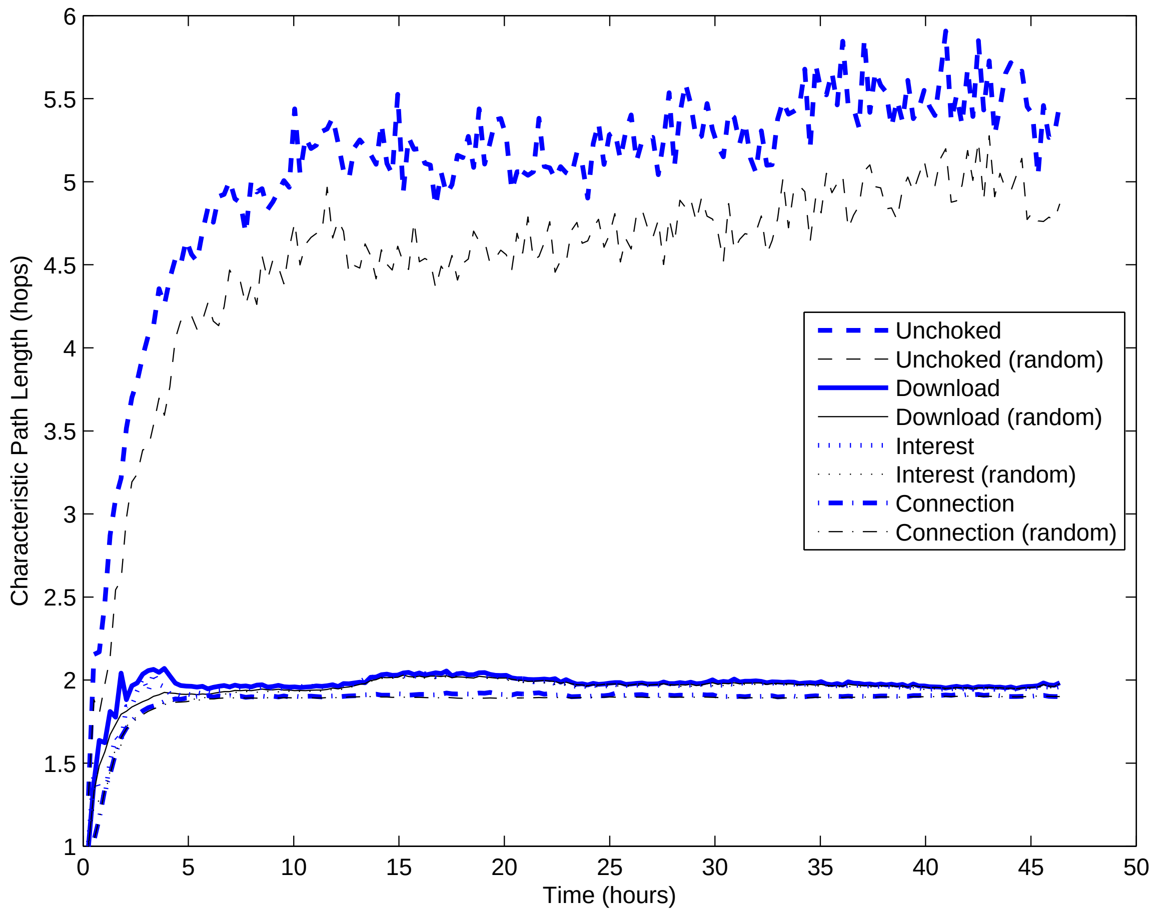

Figure 6.5: The characteristic path length during the experiments. Also shown are those of a similar-sized random graph.

has a sharp decline during the startup stage as the size of the graph increases. Once all the peers have joined the system there is some further decrease in the coefficients of all but the Unchoked graph during the transient stage. Through the end of the transient stage there are some further small oscillations in the Interest and Download graphs, until all settled into a steady stage after approximately 20 hours.

Although at first it seems that there is some clustering present in Figure 6.6, especially in the graphs of Connection, Interest, and Download peers, further investigation shows that is not the case. Figure 6.7 shows the clustering coefficients of the graphs when compared with (divided by) that of a similar sized random graph (same node and edge restrictions), which is not expected to have any clustering at all. Here we see that there is some clustering during

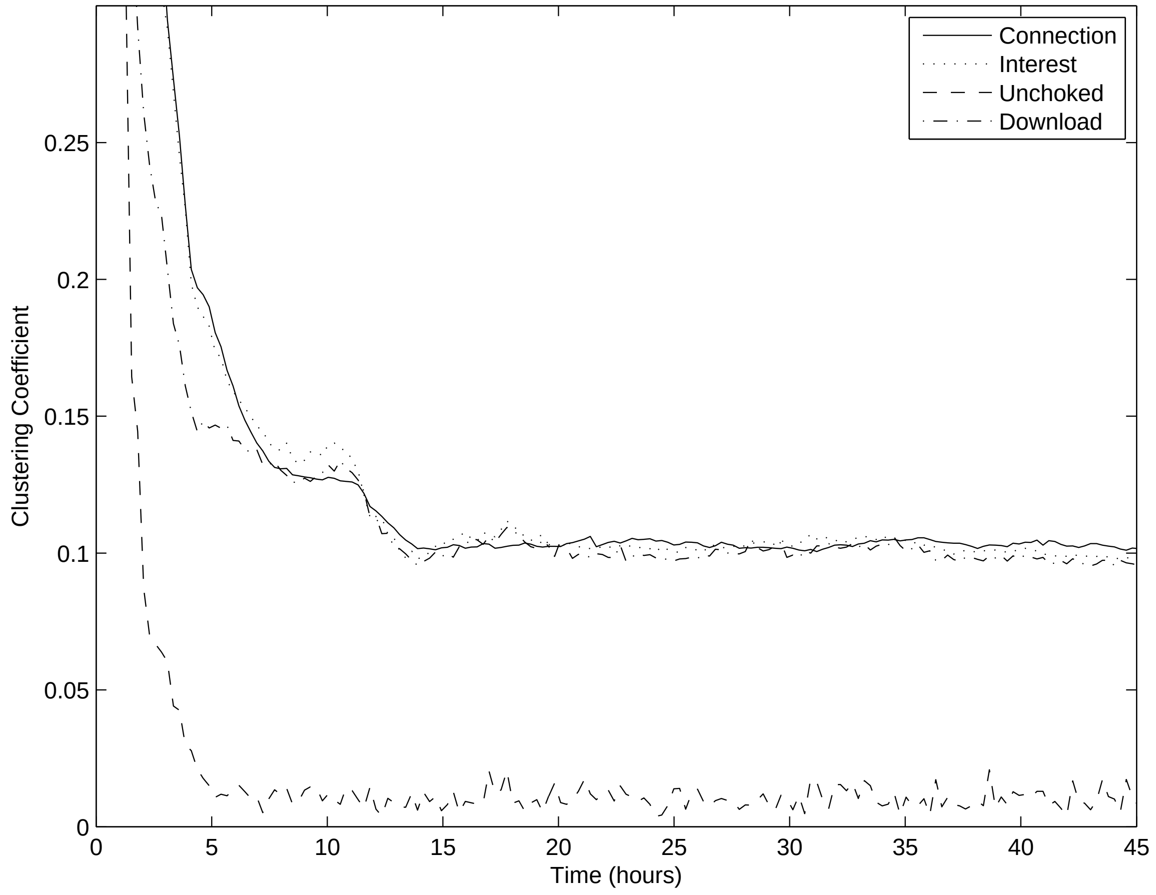

Figure 6.6: The clustering coefficient during the experiments.

the startup stage which begins to decrease once all the nodes have joined the system. The Unchoked graph has no clustering through the rest of the experiment, while the clustering of the other graphs reduces more slowly through the transient stage. In the steady stage, all graphs have almost no clustering. The increased noise in the comparison of the Unchoked graph with a random graph is due its relatively tiny clustering coefficient.

6.2 Connectivity Matrix

To compare with the results from previous papers [2,30], we present the connectivity matrix of peer connections during the experiment. The connectivity matrix is a scatter plot, where a point at location (i, j) in the plot refers to the fact that peer i is connected to peer j. The

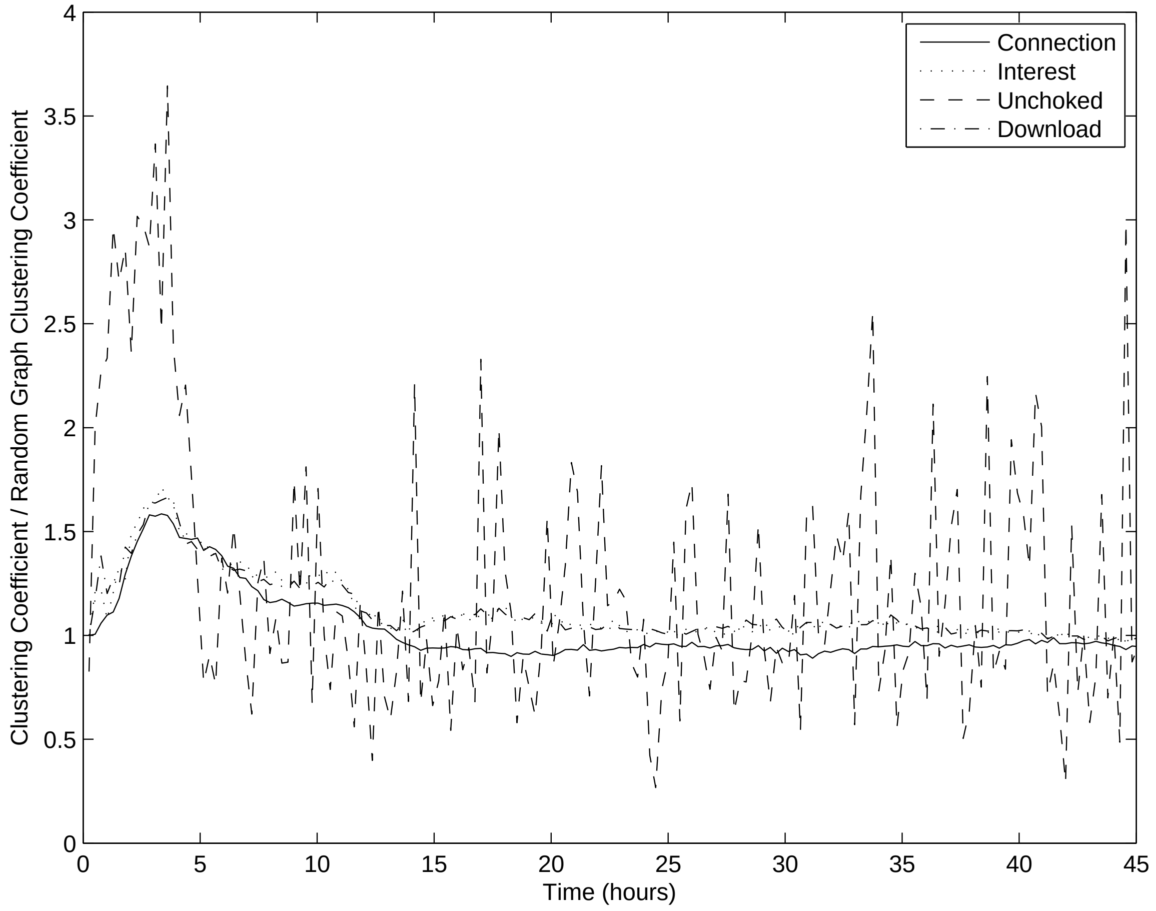

Figure 6.7: The clustering coefficient during the experiments compared with the clustering coefficient of a similar random graph.

peer index i is created by sorting the peers by their joining time.

Figure 6.8 shows the connectivity matrix formed after 4 hours at the end of the startup stage when most of the peers have joined the swarm. The fan-out shape from the lower left to upper right corner of the matrix occurs due to the early peers filling their 80 neighbor limit and refusing later connections, and is very similar to previous results [2] shown in Figure 6.9. However, there is some additional connectivity between early and late peers, which is due to some of the early peers being the fastest downloaders and having already completed their downloads. Once they become seeds they disconnect from other seeds in the system, thus freeing up neighbor slots for later peers. The previous results shown in Figure 6.9 were taken before any peers became seeds, and so do not show this feature.

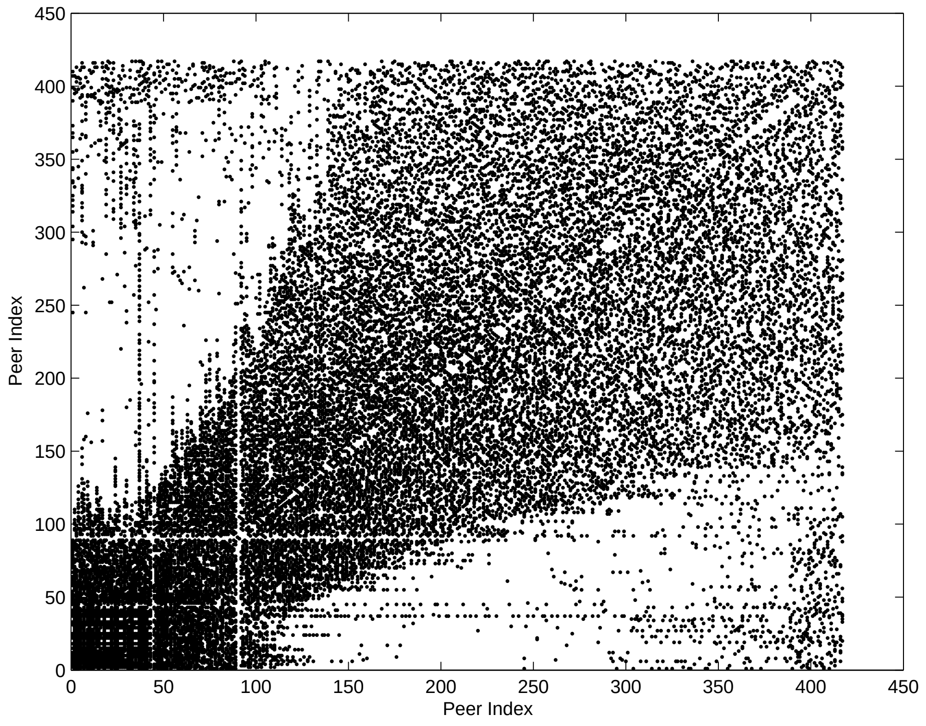

Figure 6.8: The connectivity matrix at hour 4.

Although Figure 6.8 does match well with the previous results at the early stages of the experiment, we now proceed further into the experiment to see how the connectivity matrix evolves. Figure 6.10 shows the connectivity matrix 4 hours further into the experiment in the middle of the transient stage, at which point some peers have left and new peers have joined the system. The matrix is now much more random, with many early peers having lost connections to leaving peers and so allowing connections from many late peers, though the fan-out is still visible in the lower left corner. Figure 6.11 goes further to 16 hours into the experiment, where the connectivity matrix becomes an almost completely random scattering of points, and the fan-out in the lower left is almost not visible. The connectivity matrix has now reached a steady stage, as shown by the similarity of Figures 6.11 and 6.12.

The experiment we ran with no limit on the number of neighbors a peer could have gives almost identical results to that in Figures 6.1 through 6.7, but differs from the connectivity matrix shown in Figure 6.8. In that experiment, the connectivity matrix throughout the entire experiment was completely random (similar to Figure 6.12), as the fan-out shape in Figure 6.8 is due only to the limit on the number of neighbors a peer can have.

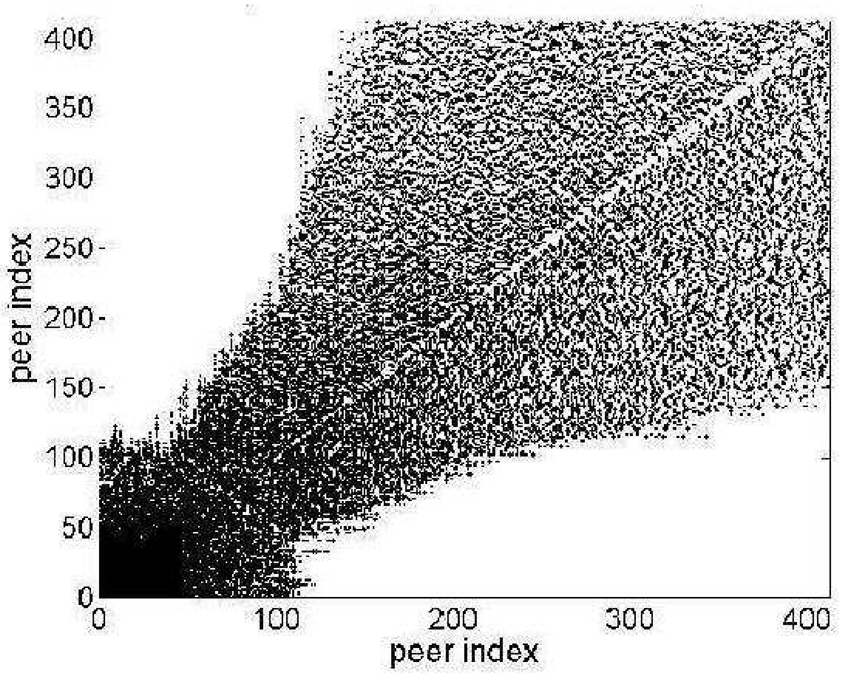

Figure 6.9: The connectivity matrix from the previous results by Al-Hamra et. al.

6.3 Impact of Churn

To further evaluate the impact of churn on the network topology, we varied the amount of churn at certain points in the system by grouping some of the peer departures and arrivals together[1]. This experiment was the same as in the previous section on the piece population in the presence of churn, and so the resulting population of seeders and leechers in the system is shown in Figure 5.14. As before, the increased churn occurs when the number of seeds decreases rapidly; for example, from 20 to 25 hours, 40 to 50 hours, and 60 to 70 hours. Since the number of peers is limited, the periods between these increased churn periods exhibit a state of decreased churn as compared with the previous experiment. We find that this varying churn had almost no effect on the power law exponent or the characteristic path length of the resulting graphs, which are identical to the previous results shown in Figures 6.4 and 6.5.

Figure 6.13 shows the clustering coefficient for the four networks in the experiment with

[1]The previous experiment also had churn, but the amount of churn was steady throughout the experiment.

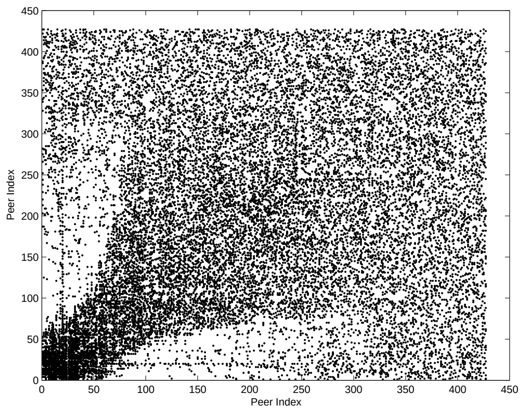

Figure 6.10: The connectivity matrix at hour 8.

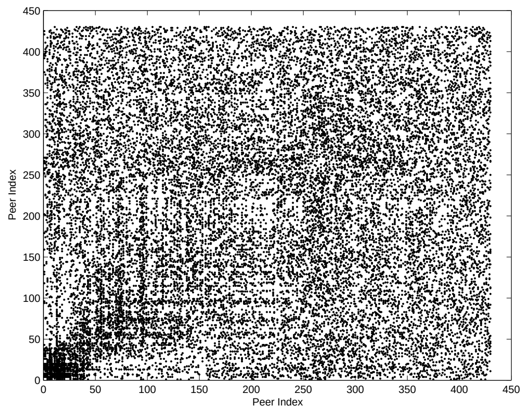

Figure 6.11: The connectivity matrix at hour 16.

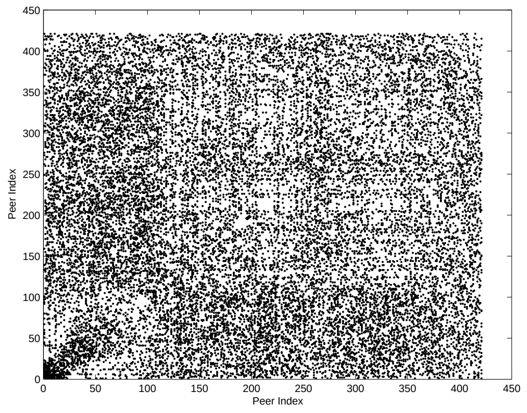

Figure 6.12: The connectivity matrix at hour 32.

varying churn. During the initial stage, it is very similar to Figure 6.6, decreasing rapidly as the peers enter the system. However, after the initial stage (i.e., after 4 hours), the effect of the varying churn can be clearly seen on the Connection, Interest, and Download networks, causing their clustering coefficients to oscillate. Interestingly, the clustering coefficient increases during the periods of light churn and decreases during the periods of heavy churn. Although the varying churn continues throughout the experiment, the oscillations in the clustering coefficients of these graphs are greatly reduced after 50 hours.

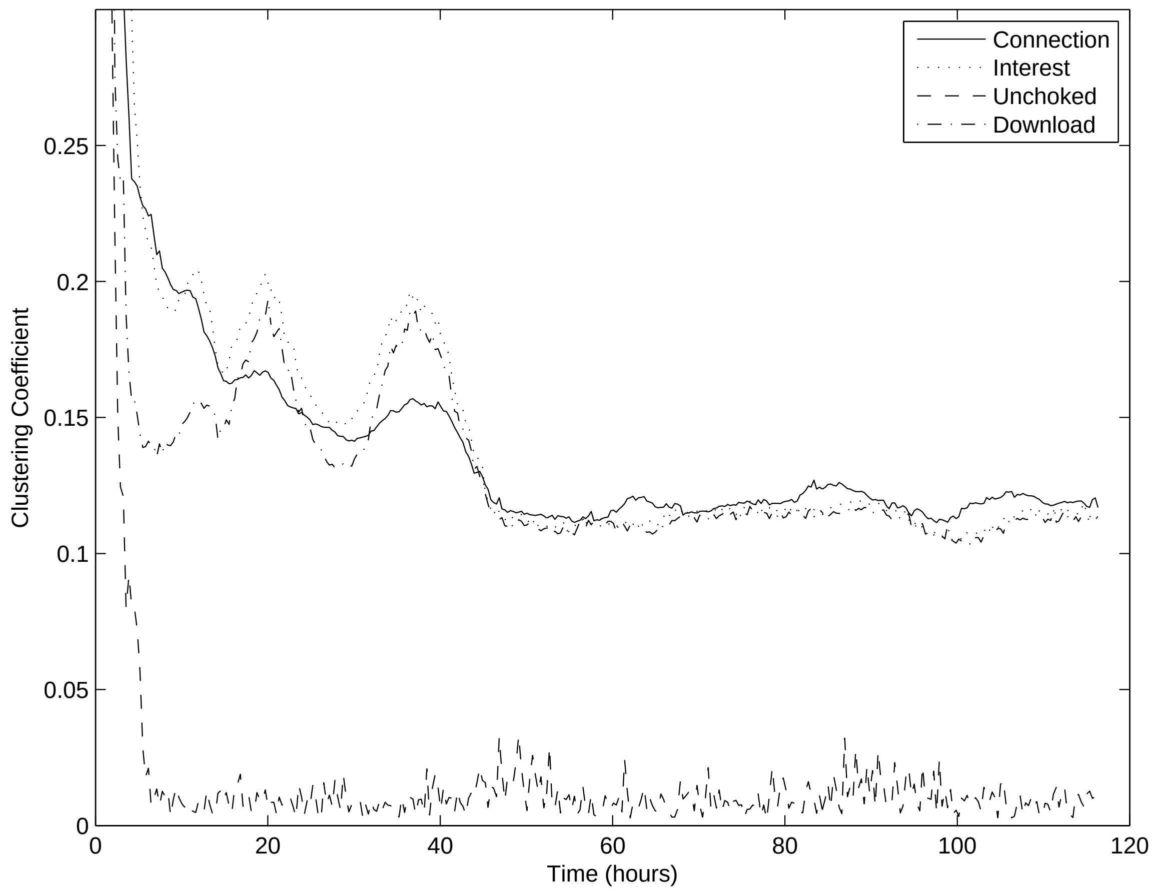

Figure 6.13: The clustering coefficient during the experiment with varying churn.

Chapter 7

Small-World Enhancement

In Chapter 6, we found no evidence of persistent clustering in any of the four BitTorrent networks we studied, and therefore no small-world networks are possible. It is known that small-world networks are efficient for spreading information [11, 12, 21]. A previous study [23] has also conjectured that BitTorrent's efficiency partly comes from the clustering of peers.

It is thus interesting to see whether BitTorrent networks can be made to cluster while maintaining a short characteristic path length, and so become small-world. We now demonstrate a possible enhancement toward this direction through both theory and practical implementation. In section 7.1, we present a theoretical attempt at increasing the small-world characteristics of BitTorrent networks. In section 7.2 we show how this can be done in practice, and measure its effectiveness through simulation and experiment.

7.1 Theory

To introduce small-world characteristics into a BitTorrent-type network, we will first show a regular graph construction that maximizes the possible clustering coefficient in section 7.1.1. We will then show the drawback of this construction, its much larger diameter, in section 7.1.2. Finally, we will add some randomness to the construction, which will make it small-world, in section 7.1.3.

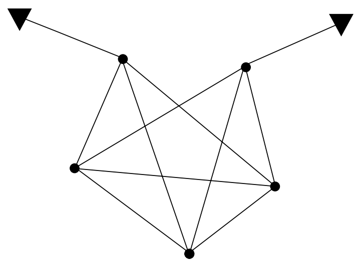

Figure 7.1: An example of one of the 5-cliques from a cycle, the arrows indicate connections to neighboring cliques.

7.1.1 Maximum Clustering Coefficient

Since most existing small-world graphs are sparse, and the BitTorrent networks we are considering are quite dense, it is not clear that creating a small-world network in BitTorrent is even possible. Therefore we start our investigation by determining the maximum possible clustering coefficient for our BitTorrent networks. We will focus on the Connection Network, which is a superset of the other three networks.

To create a small-world network containing peers with a known maximum degree, we first attempt to maximize the clustering coefficient of a regular graph of these peers. Our instinct is to create a series of cliques (since they have a perfect clustering coefficient of 1), each with size equal to the maximum node degree. A single edge can then be removed from each clique (to maintain regularity), and the endpoints of the removed edge connected to neighboring cliques to connect the graph. This results in a cycle of k identical *n-cliques*, where n is the maximum node degree, and k is given by $k = N/n$ (where N is the total number of peers, and assuming for now that k is an integer). Figure 7.1 shows an example of one of the n-cliques from the cycle for $n = 5$.

As the clustering coefficient of the graph is an average over all nodes, and each n-clique is identical, it will be sufficient to calculate the clustering coefficient of a single n-clique. Each n-clique contains two types of nodes: $n-2$ interior nodes (i-nodes) connected only to neighbors in the same n-clique, and 2 exterior nodes (e-nodes) that have a single connection to another n-clique.

Since the clustering coefficient of a node is a measure of how many triangles include the node, it is sufficient to consider only how many triangles are lost from a perfect clique (of $(n-1)(n-2)/2$ triangles) by removing the single edge to connect to neighboring n-cliques. The i-nodes lose only a single triangle when the edge is removed, so their clustering coefficient is

$$C_i = \frac{\frac{(n-1)(n-2)}{2} - 1}{\frac{(n-1)(n-2)}{2}} = 1 - \frac{2}{(n-1)(n-2)} \tag{7.1}$$

The e-nodes lose a triangle for each node that was connected to the node that is no longer a neighbor due to the missing edge, of which there are $n-2$. So their clustering coefficient is

$$C_e = \frac{\frac{(n-1)(n-2)}{2} - (n-2)}{\frac{(n-1)(n-2)}{2}} = 1 - \frac{2}{(n-1)} \tag{7.2}$$

Averaging over all the nodes in the n-clique gives the clustering coefficient of the n-clique, and also of the entire graph

$$CC(G) = \frac{(n-2) * C_i + 2 * C_e}{n} = 1 - \frac{6}{n(n-1)} \tag{7.3}$$

This result is independent of the total size of the graph, and so of the number of n-cliques used. It also approaches 1 as the size of the n-cliques increases. For BitTorrent, which has a default maximum node degree of 80, the clustering coefficient is very close to 1 (0.9986 for $n = 80$).

7.1.2 Expanding Diameter

The clustering coefficient resulting from our construction is large, but the diameter and characteristic path length of the graph are also large, making it far from small-world. It takes 3 hops to get through a single clique, and the worst case is having to go half way around the cycle of k n-cliques; hence, the maximum diameter is given by approximately $\frac{3k}{2}$ (7.5 for a BitTorrent graph of 400 nodes with an n-clique size of 80).

We can estimate the characteristic path length by considering only the distances of i-nodes from other i-nodes (since there are many more of them than there are e-nodes). For each i-node there are $n-1$ nodes at distance 1 (in the same n-clique), $2n$ nodes at distance 3 (one n-clique away), $2n$ nodes at distance 6, etc.... The sum of the distances to all i-nodes is then

$$(n-1) + 6n \sum_{j=1}^{\frac{k-1}{2}} j = n - 1 + 3n\frac{k-1}{2}\left(\frac{k-1}{2} + 1\right) \tag{7.4}$$

The characteristic path length of the graph can be calculated by using an approximation for large values of n and dividing by the number of nodes

$$CPL(G) \approx \frac{n + \frac{3}{4}n(k-1)(k+1)}{nk} = \frac{1 + \frac{3}{4}(k^2-1)}{k} \tag{7.5}$$

For a BitTorrent graph with $n = 80$ and 400 nodes ($k = 5$), the characteristic path length will be 3.8, which is almost twice that of a similarly sized random graph.

7.1.3 Forming the Small-World

To minimize the characteristic path length without disrupting our highly clustered peers, we use the same technique as Watts and Strogatz [32] and add controlled amounts of randomness to the network. We modify the regular n-clique graph by randomly removing a small number of n-clique edges and adding back new randomly created ones. In our construction, the new random edges are restricted to be links between pairs of n-cliques.

Figure 7.2 shows the result of adding varying amounts of randomness to the regular n-clique graphs for a few sizes of graphs. We again use a clique size of 80, which is the default maximum node degree of BitTorrent graphs. The randomness varies from 0 (no edges replaced, completely regular) to 1 (all edges replaced, making the graph almost completely random). We observe that the graphs become small-world when approximately 1 to 3% of the edges are randomly replaced. At this point the clustering coefficient still maintains 90% of its original value, while the characteristic path length has dropped very near to that of a completely random graph.

7.2 Implementation

To realize the theoretical results in practical BitTorrent software, we have implemented a simple modification to the BitTorrent tracker, which is described in section 7.2.1. We believe this is the best, and possibly only, type of change that can be made to BitTorrent's functionality. The tracker is easily modifiable by any data distributor, whereas the vast diversity of BitTorrent clients that would need to be changed for a client modification make that type of change much more difficult. We then analyse the effect of our new tracker through both simulation, in section 7.2.2, and experimentation, in section 7.2.3.

7.2.1 Tracker Modification

The modified tracker assigns a number to each newly arrived peer indicating the n-clique to which it belongs. If all the n-cliques are full (or there are no n-cliques), the peer will receive a new n-clique number one larger than the largest so far. Otherwise the peer receives the

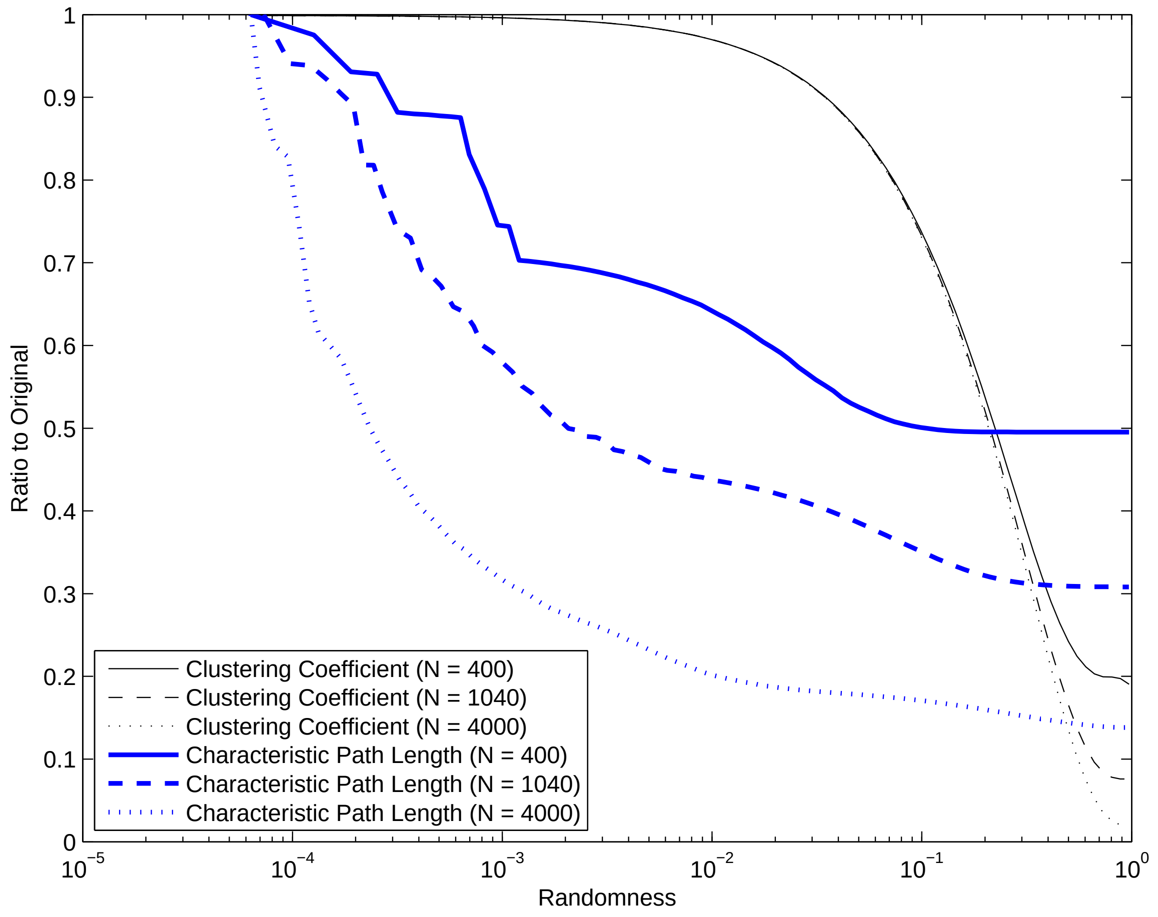

Figure 7.2: The change in the clustering coefficients and characteristic path lengths of the n-clique graphs with varying amounts of randomness added. The variables are shown as ratios to the values when no edges are replaced. The original clustering coefficients were all 0.999, while the original characteristic path lengths were 3.64, 6.25 and 15.8 for the 400, 1040 and 4000 node graphs respectively.

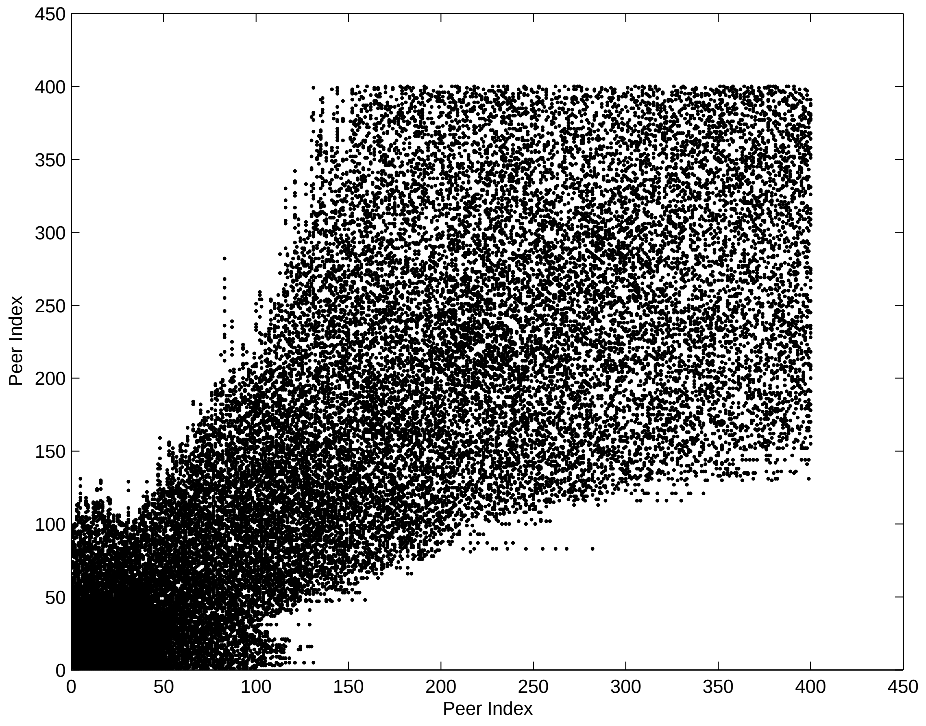

Figure 7.3: The simulated connectivity matrix after 400 peers have joined the swarm.

number of the largest currently unfilled n-clique. When choosing a list of other peers to return to the current peer, the n-clique number of the peer will be considered. The tracker will first choose a small fixed number of peers randomly from n-cliques that do not contain the peer. This small number of peers will be the control on the randomness referred to in section 7.1. The remaining peers (the majority) will be randomly chosen from those in the same n-clique as the current peer.

There are two new configuration parameters for this tracker. The first, which we call `random-peers`, is the fixed number of peers that will be randomly chosen from other n-cliques for each request. The second, which we call `clique-size`, is the maximum size of n-clique that the new tracker will allow. Once all n-cliques reach this size, the next peer to join will be added to a new empty n-clique.

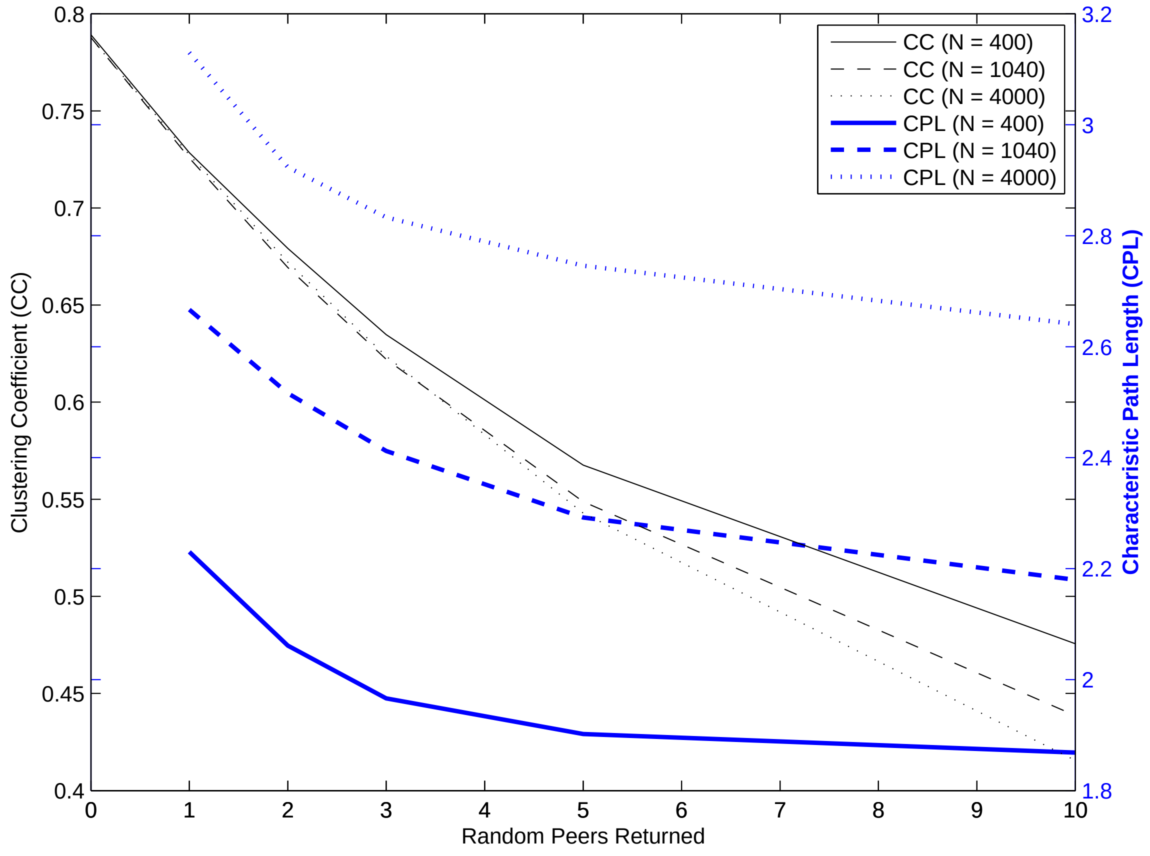

Figure 7.4: The simulated small-world tracker results for three sizes of graphs (`clique-size` is kept constant at 80).

7.2.2 Simulation Results

We first present some results from a customized discrete event-driven simulator of BitTorrent that we have created in MatLab. We have verified the correctness of the simulator by first using a standard tracker implementation that returns a random set of peers in each request, and comparing the results with our previous experimental results. There are 400 peers arriving over a period of 4 hours, followed by 2 hours during which the peers make further requests from the tracker to satisfy their minimum peer requirements. Figure 7.3 shows the connectivity matrix after 6 hours, which can be compared with Figure 6.8 from our previous experiment. Other than the connections in Figure 6.8 due to peers becoming seeds (which we did not simulate), the results are almost identical.

For the modified tracker, Figure 7.4 shows the resulting clustering coefficients and characteristic path lengths for three sizes of swarms with `random-peers` values from 0 to 10. The characteristic path length for 0 `random-peers` does not appear, as it is infinite due to the disconnected nature of the graph. Though we could implement fractional values for the `random-peers` parameter to better explore the possible settings (a value of 1 is already 2% of the peers returned, which is a large amount of randomness in figure 7.2), we instead decided to keep our tracker as simple as possible. With a value of 1 we were able to achieve a high enough clustering coefficient that we would not want to use a lower value anyway, for fear of unduly increasing the characteristic path length.

Compared with the previous experimental results, there are gains in the clustering coefficient of the graph, but they are not as dramatic as the theory in section 7.1 suggests (i.e. they are not almost 1 for zero randomness). This is due to the limits on the number of connections that a peer can initiate in BitTorrent (default is 40), which prevents the peers from forming a complete n-clique. Since we are trying to only modify the tracker, we cannot adjust the limit on the number of connections at which a BitTorrent client stops initiating more.

However, we can reduce the `clique-size` small-world tracker configuration parameter so that more connections are made within the clique. Figure 7.5 shows the result of changing the `clique-size` parameter, for swarms of 400 nodes only. By decreasing the size of the n-clique, the n-cliques become more populated with edges, increasing the clustering coefficient to almost 1 for the case of 40-cliques with no randomness. From Equation 7.3, we calculate that the theoretical maximum possible clustering coefficient for n-cliques of size 40 is only slightly reduced to 0.9962 (from 0.9986 for size 80).

7.2.3 Experimental Results

We further confirmed the effectiveness of the modification experimentally. The experiment was run as described previously in Chapter 4, with the only difference being the use of the modified small-world tracker. The modified tracker used a `random-peers` value of 1, and a `clique-size` value of 40. The population of peers in the system during the experiment was identical to the previous experiment shown in Figure 6.1.

Figure 7.6 shows the characteristic path length for the four networks. As expected, there is an increase in the characteristic path length due to the regular construction of the graph imposed by the tracker, as compared with the previous results in Figure 6.5. The increase

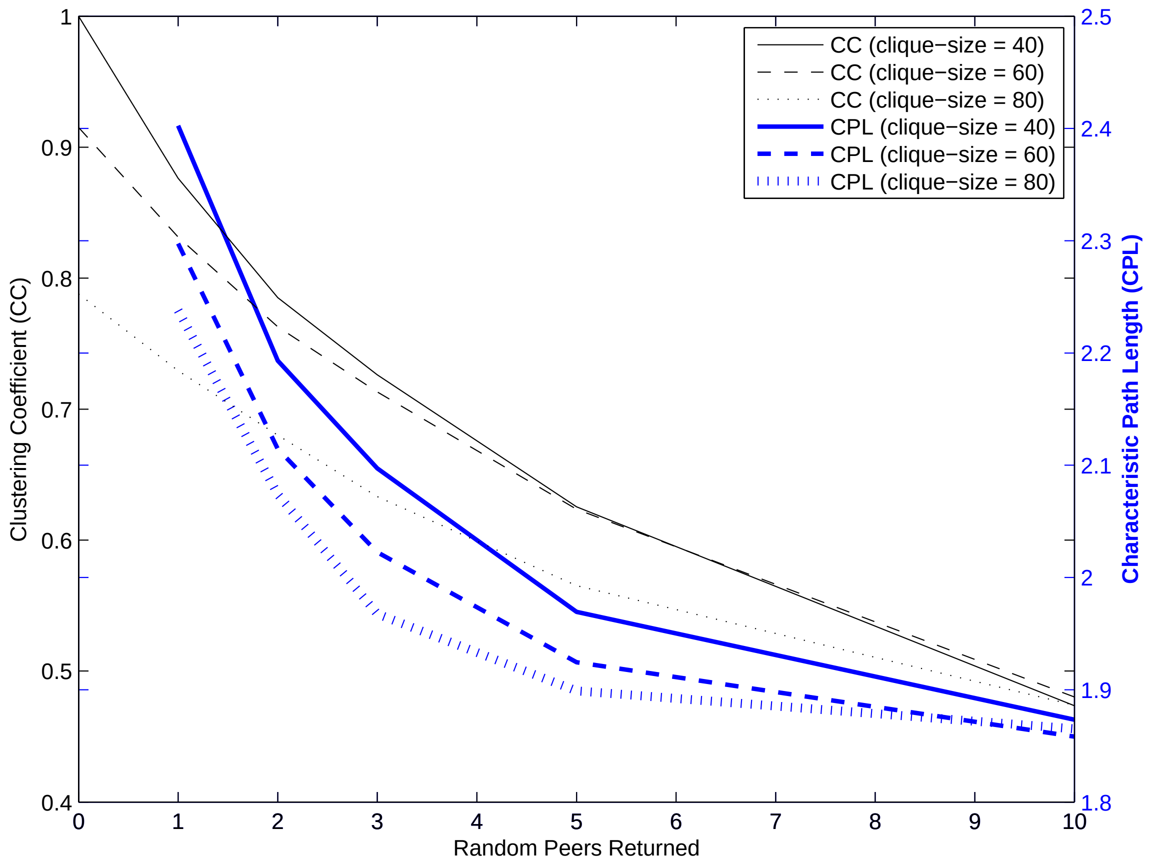

Figure 7.5: The simulated small-world tracker results for three values of the `clique-size` configuration parameter (the number of nodes is kept constant at 400).

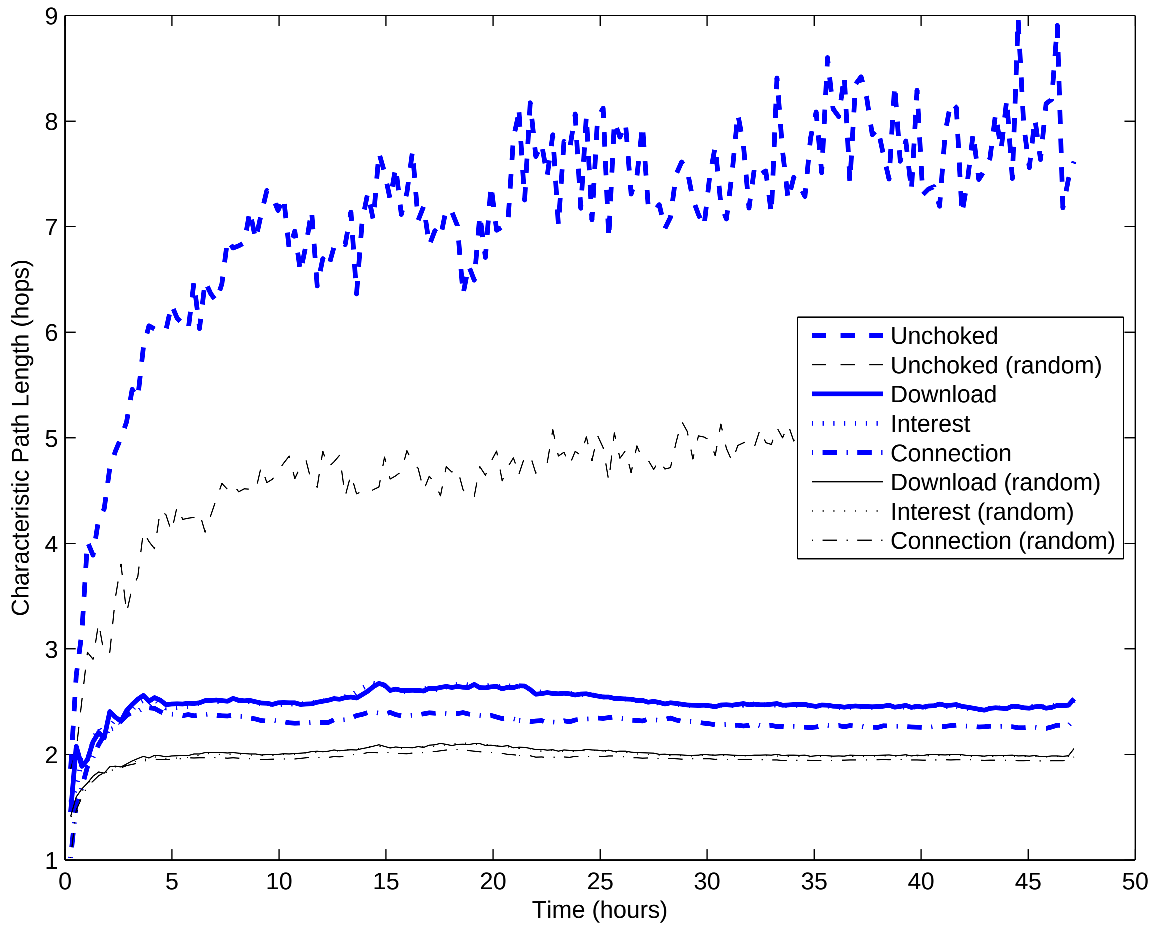

Figure 7.6: The characteristic path lengths during the small-world tracker experiment, also shown are those of a similar-sized random graph.

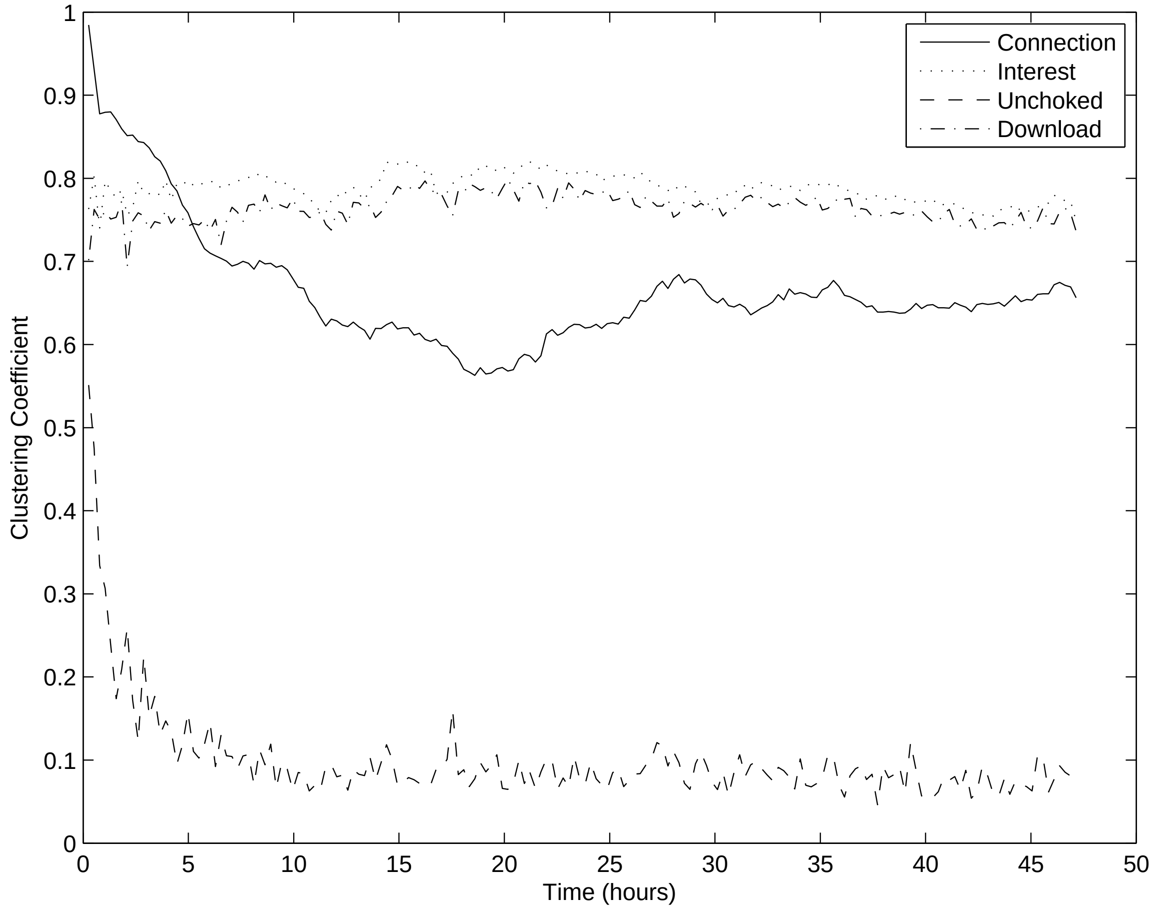

Figure 7.7: The clustering coefficients during the small-world tracker experiment.

is on the order of 10 to 20%.

Figure 7.7 shows the clustering coefficient for the four networks in the experiment. Compared with the previous results (shown in Figure 6.6), the clustering coefficient has increased by a factor of 7 or 8. To determine if there really was clustering present, we again compared the clustering coefficients of the graphs with that of a similarly sized random graph in Figure 7.8. Here we see that there is definite clustering throughout the experiment in all four of the networks. These results are a dramatic increase over the amount of clustering present in the previous experiments, shown in Figures 6.6 and 6.7.

Figure 7.9 shows the connectivity matrix at hour 4 of the experiment, after all the original peers have joined the swarm. This connectivity matrix is constructed differently

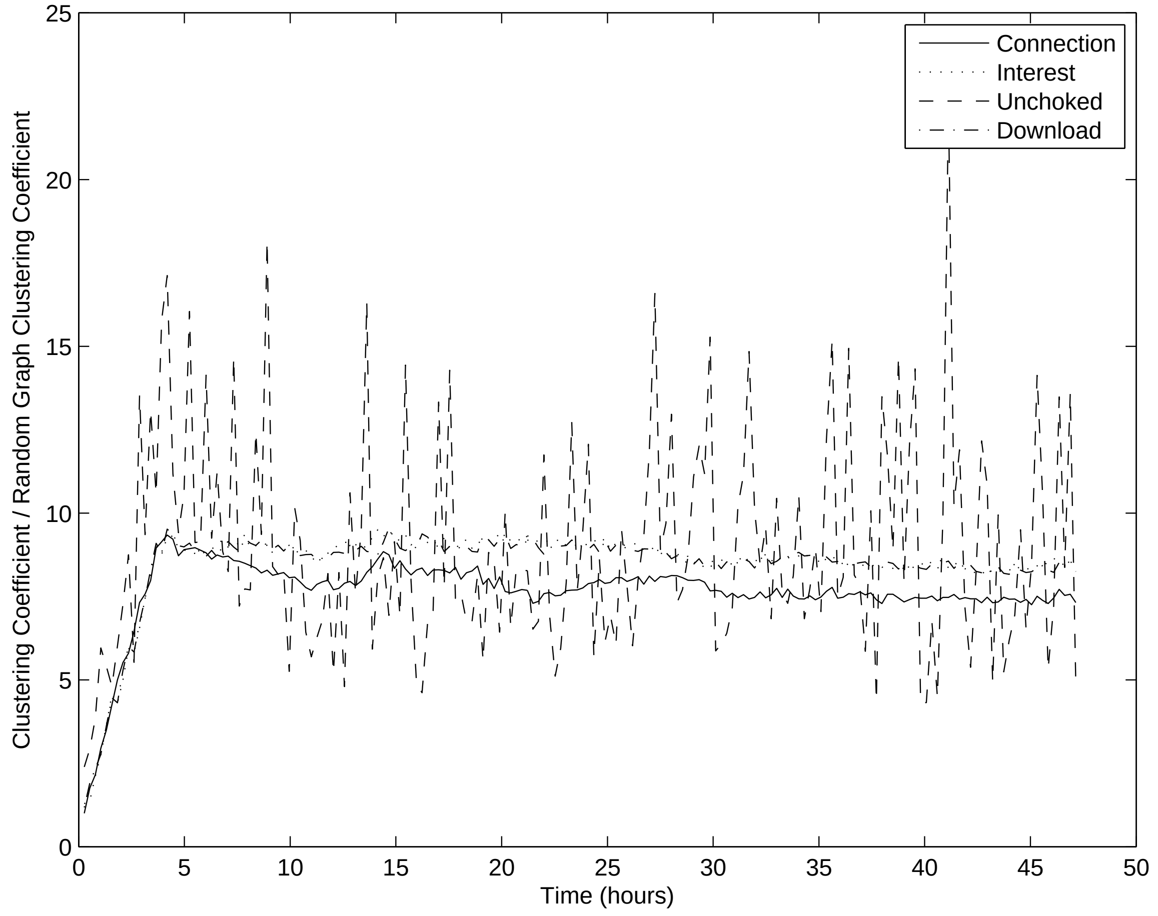

Figure 7.8: The clustering coefficients during the small-world tracker experiment, compared with the clustering coefficient of a similar random graph.

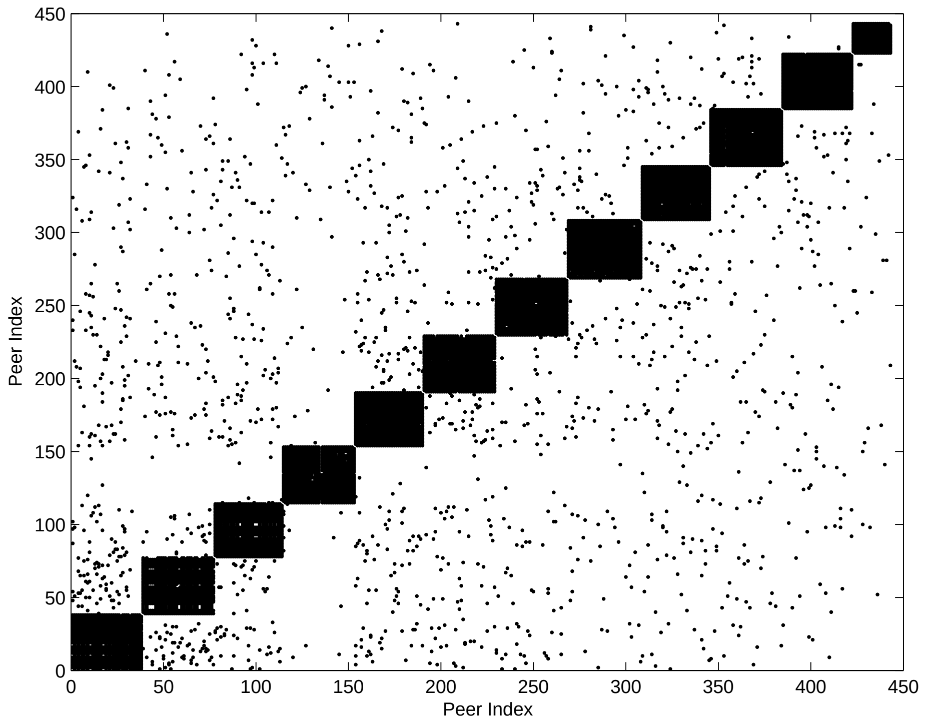

Figure 7.9: The connectivity matrix at hour 4 of the small-world tracker experiment.

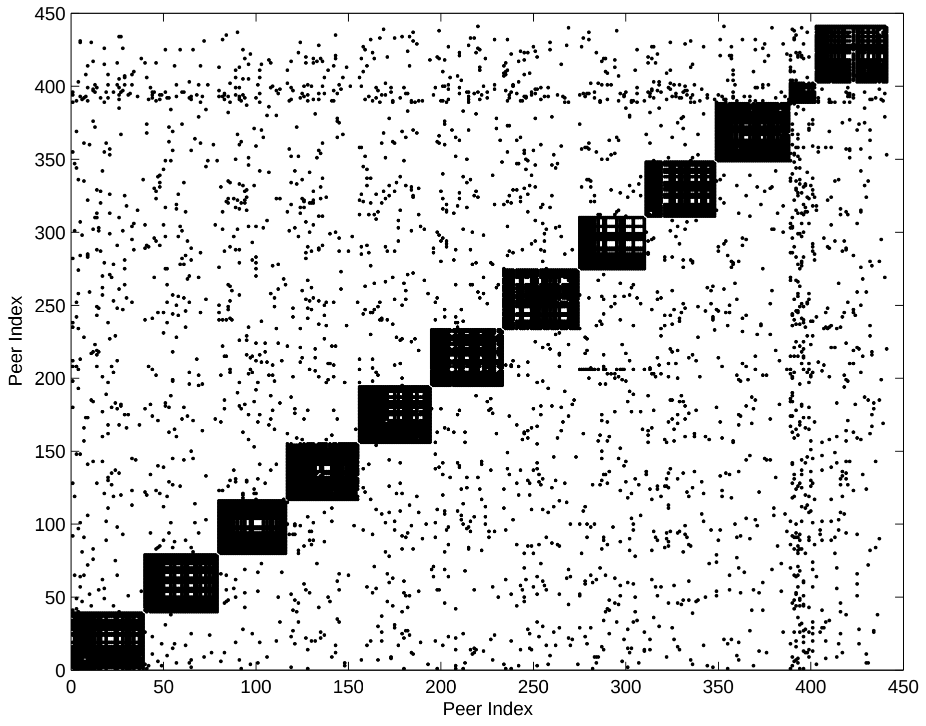

Figure 7.10: The connectivity matrix at hour 32 of the small-world tracker experiment.

than previously: rather than ordering the peers by their arrival times to get their peer indices, we instead sort them by the clique to which they belong.[1] This shows the cliques that we have constructed more clearly, as can be seen by the tight boxes of peers in the figure. The matrix is now quite different than the previous results shown in Figure 6.8. The formation of the n-cliques is very clear, with some random connections to other n-cliques also present.

We find that the n-cliques are still clearly formed throughout the entire experiment, although they develop holes due to the seeds dropping connections to other seeds. This is shown, for example, by Figure 7.10, which is the connectivity matrix 32 hours into the experiment.

[1]We also attempted this with the previous results and got an almost completely random connectivity matrix similar to Figure 6.12.

Chapter 8

Discussion

8.1 Piece Population

As we have seen, the snapshots of piece populations roughly follow normal distributions with means near half the number of downloaders. Their block variations were small, and the tails expected for the larger swarms were small as well, indicating the effectiveness of the rarest-first policy at replicating pieces uniformly using only the local information available to each peer. Note that this is the same conclusion reached by Legout et al. [22], but in this case we did not rely on the somewhat questionable definitions of *entropy* and *peer availability* to reach that conclusion, as we have the global piece data needed.

The population evolution over time showed similar shapes. In addition, as the swarm progressed out of the transient stage and into the steady stage, we have seen a clear progression in the shape of the piece population. This progression leads to a more ideal shape, as the mean increased and the block variation decreased.

In the first PlanetLab measurement the population of the early stages in a swarm's lifetime was quite poor at matching the ideal narrow distribution. This is attributable to the poor availability of some pieces in the system, which have not yet been uploaded by the initial seed. This changed very quickly after a single copy of the file was available, indicating that the policy works well once this startup stage is complete. Once the last piece is available in the system, the swarm quickly replicates the last pieces uploaded so that they can "catch up" to the replication that earlier pieces have already had.

In the second PlanetLab measurement, we presented a more realistic deployment of BitTorrent. We saw how increased churn can increase the width of the piece distribution, as the block variation was shown to increase during these periods of high churn. This is probably due to BitTorrent's policy of new peers selecting pieces at random for a short period to bootstrap themselves in the swarm more quickly. However, once the periods of churn were over, we saw the block variation again decrease to its near-optimal value very quickly as the rarest-first policy took over and effectively narrowed the distribution. We also ran an additional experiment using a random piece policy for comparison, and found that it resulted in a block variation that was larger by an order of magnitude. This shows the effectiveness of the rarest-first piece policy in comparison with its predecessor, and the reason why it was chosen as a replacement piece selection policy.

In summary, the rarest-first policy employed by BitTorrent is successful, both in moving quickly to a narrow distribution, and in maintaining and even improving that distribution as the swarm progresses. There are concerns though, as both the early startup stage and some of the larger swarms show increased width due to the limited piece availability and the limited knowledge of the system a peer has. The first is not easy to improve, as any policy will be hampered by the limited selection of pieces available in the system at the early stages of a swarm. However, the tails seen on some of the larger swarms' population distributions could be improved upon by increasing the amount of knowledge each peer has of the system. For example, when communicating the pieces each peer has to its neighbors, extra information could be included such as the rarest pieces seen by the peer or the number of copies each piece has in its limited local view. This new information, combined with the knowledge a client already has of its neighboring pieces, is believed to be enough to reduce the size of the tail on the population distribution at very little cost in extra communication.

8.1.1 Future Work

More measurements could be made, focussing on much larger swarms, to determine the size that population tails can grow to. More PlanetLab experiments could also be run consisting of a much larger number of peers joining after the startup stage, and progressing through a transient to a steady stage. With these results, modifications to the rarest-first policy could be made to determine if it is effective in reducing the size of the tail.

Furthermore, piece-level models describing the piece population of a torrent swarm are needed, which will better explain the pros and cons of the rarest-first policy, as well as facilitate these modifications. Our preliminary modeling results can be found in our technical report [14].

8.2 Network Topologies

We also presented an experimental study of the characteristics and evolution of a comprehensive set of BitTorrent network topologies, as well as a possible enhancement to BitTorrent to make it a small-world.

Our most important finding was that the startup stage of a BitTorrent system is not predictive of the overall performance of the system. In order to fully examine a BitTorrent swarm, long-term experiments that examine the changes that occur in the later stages of the swarm are needed. This was clearly demonstrated by the differences that we found between the steady stage matrix connectivity and the connectivity that was previously reported to occur early in the lifetime of a BitTorrent swarm [2]. The differences are almost certainly due to the evolution of the swarm over time: as peers become seeds they break connections with other seeds, and when peers leave they free up neighbor slots for new peers. Neither of these aspects were previously considered. Furthermore, peers use the tracker to form new random connections to peers. We found that, on average, a peer will get a new random peer list from the tracker at least once during its time in the swarm. This behavior was not considered at all in previous studies.

Our experimental results showed that the network of peers that unchoke each other is scale-free, exemplified by a power law distribution of node degrees. We found that the Unchoked graph has a power law exponent of approximately 2 in all of the experiments, independent of time and changes to the amount of churn or the limit on the number of neighbors. This scale-free nature has also been found in other peer-to-peer systems [24]. Since scale-free graphs are resistant to random attacks (in this case, churn), the resulting graphs are more robust than graphs without this property[1]. This robustness was observed previously in BitTorrent networks [2], but no explanation for its existence was given. We believe that the robustness to churn of the scale-free Unchoked graph, the graph connecting peers for downloading, is responsible for the efficient use of upload bandwidth previously observed in BitTorrent systems [20, 29].

We found that the characteristic path lengths and clustering coefficients of the Connection, Interest, and Download graphs are very similar to random graphs, and that there is no evidence of clustering or small-world characteristics. Other than the scale-free degree

[1]The vulnerability of scale-free graphs to targeted attacks is not an issue in BitTorrent because the tracker prevents the graph from becoming disconnected.

distribution, we found that the Unchoked graph is also nearly random after a short initial period. We showed quantitatively that there is a small amount of clustering in the Unchoked network during the startup stage, confirming the previous qualitative evidence [23]. However, after this short period we found no evidence of clustering in any of our experiments, which precludes the presence of a small-world network. In addition, we did find some interesting increases in the clustering coefficient during periods of light or no churn in the current BitTorrent. We believe that it is this churn, in combination with the random list of peers returned by the tracker (designed to create a random graph), that is preventing clustering from occurring.

Small-world characteristics are desirable for efficient information distribution [11,12,21], and previous studies have also conjectured that BitTorrent's efficiency partly comes from the clustering of peers with similar bandwidth [23]. We believe that this can be an interesting venue for improving BitTorrent's performance. We therefore presented a theoretical framework for making BitTorrent small-world, together with a practical implementation. Our implementation makes minimal changes to BitTorrent trackers only, and the preliminary results show that the simple modification created a dramatic increase in the amount of clustering, at the expense of a slightly increased diameter. Although the tracker controls only one of the four networks in BitTorrent, i.e. the Connection network, we showed that our modification also introduces small-world characteristics to all the other networks, including the important Unchoked network.

8.2.1 Future Work

Though we have introduced the desired small-world characteristics to all 4 of the BitTorrent networks we considered, there are some additional factors to consider and future work needed.

First, and foremost, the efficiency of the small-world swarms needs to be tested to determine what effect these changes have had. Though small-world networks have been shown to be efficient for peer-to-peer file distribution, this efficiency needs to be quantified and compared with the original BitTorrent tracker implementation.

Peer departures have not been adequately considered by our current modification, and will probably leave cliques short of their target size. The experiment only considers an increasing and steady state network size, but not what happens when the network size decreases. Additional tracker modifications may be needed to consolidate small cliques, or to move peers from one clique to another. Also, seeds limit the number of connections possible in a clique, since they will not connect to each other, as shown in Figure 7.10. This will reduce the clustering coefficient of the graph.

Using a smaller clique size causes the peers to re-request more peers from the tracker more frequently. This is due to the peer trying to maintain at least 40 peer connections. These re-requests will have the side-effect of increasing the number of random connections in the graph due to the single random peer returned in every request. Using a larger `clique-size` value may help, or the tracker could only choose a random peer once for each peer (i.e. check if a random peer has already been given to this peer, if so and the random peer still exists then return it, otherwise generate a new random peer). This fixed random peer solution would also prevent malicious peers from gaining access to a larger number of peers outside their n-clique in an attempt to circumvent our new tracker policy.

Chapter 9

Conclusions

In this paper, we have presented measurements on the piece populations in BitTorrent swarms, and investigated the effectiveness of the rarest-first policy for piece replication from a piece distribution and evolutionary perspective. We have shown that the policy is quite effective once all pieces become available in the system, and throughout the lifetime of the swarm. However, some deviations from the ideal were apparent soon after creation of the swarm, and in some of the larger swarms studied.

We have also shown experimentally the evolving networks in a BitTorrent swarm, which we found change significantly over time. We have quantified the scale-free nature of one of the networks present in a BitTorrent swarm, that of peers unchoking each other. We found that this scale-free nature is independent of time and the changing parameters of the experiment. We have also gone beyond previous studies to show that, after the very early stages, most of the networks in a BitTorrent swarm are purely random graphs with no clustering present. Therefore, the graphs do not exhibit the small-world characteristics that have been found in many other peer-to-peer network overlays. However, the ever-present churn in a BitTorrent system may have impacted this result.

We have therefore successfully designed a tracker modification to introduce small-world networks into a BitTorrent swarm. The modification has been tested, both through simulation and experimentally, to have introduced a large amount of clustering, at the expense of only a small increase in the characteristic path length. Our changes are only to the tracker, yet we have introduced these small-world characteristics into all four of the BitTorrent networks we considered.

Reference List

[1] CacheLogic. `http://www.cachelogic.com`, 2004.

[2] Anwar Al-Hamra, Arnaud Legout, and Chadi Barakat. Understanding the properties of the bittorrent overlay. Technical report, INRIA, 2007.

[3] R. Albert, H. Jeong, and A.-L. Barabási. Error and attack tolerance of complex networks. *Nature*, 406:378–382, July 2000.

[4] D. Alderson, J.C. Doyle, L. Li, and W. Willinger. Towards a Theory of Scale-Free Graphs: Definition, Properties, and Implications. *Internet Math*, 2(4):431–523, 2005.

[5] Good settings for bittorrent from the azureus wiki. `http://www.azureuswiki.com/index.php/Good_settings`, April 2007.

[6] A.L. Barabási and E. Bonabeau. Scale-Free Networks. *Scientific American Magazine*, 288(5):60–69, 2003.

[7] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, 1999.

[8] The BitTornado website. `http://www.bittornado.com/`, 2007.

[9] B. Bollobás. *Random Graphs.* Cambridge University Press, 2nd edition, 2001.

[10] Bram Cohen. Incentives build robustness in BitTorrent. `http://bitconjurer.org/BitTorrent/bittorrentecon.pdf`, May 2003.

[11] F. Comellas, M. Mitjana, and J.G. Peters. Epidemics in Small-World Communication Networks. Technical Report SFU-CMPT-TR-2002, Simon Fraser University, October 2002.

[12] F. Comellas, J. Ozon, and J.G. Peters. Deterministic small-world communication networks. *Information Processing Letters*, 76(1-2):83–90, 2000.

[13] Cameron Dale and Jiangchuan Liu. A measurement study of piece population in BitTorrent. In *GlobeCom*, Washington, DC, November 26–30 2007.

[14] Cameron Dale and Jiangchuan Liu. Modelling piece population in BitTorrent. Technical report, Simon Fraser University, 2007.

[15] Cameron Dale, Jiangchuan Liu, Joseph Peters, and Bo Li. Eveolution and enhancement of BitTorrent network topologies. In *IWQoS*, University of Twente, Enschede, The Netherlands, June 2–4 2008.

[16] Gustavo de Veciana and Xiangying Yang. Fairness, incentives and performance in peer-to-peer networks. In *Proc. Forty-first Annual Allerton Conference on Communication, Control and Computing*, Monticello, Illinois, USA, October 2003.

[17] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang. Measurements, analysis, and modeling of BitTorrent-like systems. In *Proc. Internet Measurement Conference*, Berkeley, CA, USA, October 2005.

[18] T. Hong. Performance. In Andy Oram, editor, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, chapter 14, pages 203–241. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[19] KYK Hui, JCS Lui, and DKY Yau. Small world overlay P2P networks. *Proc. Twelfth IEEE International Workshop on Quality of Service*, pages 201–210, 2004.

[20] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Al Hamra, and L. Garces-Erice. Dissecting BitTorrent: Five months in a torrents lifetime. In *Passive and Active Measurements*, Antibes Juan-les-Pins, France, April 2004.

[21] Vito Latora and Massimo Marchiori. Efficient behavior of small-world networks. *Phys. Rev. Lett.*, 87(19):198701, Oct 2001.

[22] A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest first and choke algorithms are enough. In *Proc. IMC'06*, Rio de Janeiro, Brazil, October 2006.

[23] Arnaud Legout, Nikitas Liogkas, Eddie Kohler, and Lixia Zhang. Clustering and sharing incentives in bittorrent systems. In *SIGMETRICS*, pages 301–312, 2007.

[24] G. Liu, M. Hu, B. Fang, and H. Zhang. Measurement and Modeling of Large-Scale Peer-to-Peer Storage System. *Lecture Notes in Computer Science*, pages 270–277, 2004.

[25] S. Milgram. The small world problem. *Psychology Today*, 2(1):60–67, 1967.

[26] Di Niu and Baochun Li. On the resilience-complexity tradeoff of network coding in dynamic p2p networks. *Proc. Fifteenth IEEE International Workshop on Quality of Service*, pages 38–46, 21-22 June 2007.

[27] The PlanetLab website. `http://www.planet-lab.org/`, 2007.

[28] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. The BitTorrent P2P file-sharing system: Measurements and analysis. In *Proc. 4th International Workshop on Peer-to-Peer Systems*, Ithaca, NY, USA, February 2005.

[29] Dongyu Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In *Proc. SIGCOMM '04*, Portland, Oregon, USA, August 30–September 3, 2004.

[30] Guillaume Urvoy-Keller and Pietro Michiardi. Impact of inner parameters and overlay structure on the performance of bittorrent. In *INFOCOM*, 2006.

[31] D.J. Watts. *Small Worlds: the dynamics of networks between order and randomness.* Princeton Univ Pr, 1999.

[32] DJ Watts and SH Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):409–10, 1998.

[33] S. Willmott, J.M. Pujol, and U. Cortes. On Exploiting Agent Technology in the design of Peer-to-Peer Applications. *Lecture Notes in Computer Science*, 3601:98, 2005.

[34] R.H. Wouhaybi. *Algorithms for Reliable Peer-to-Peer Networks.* PhD thesis, Columbia University, 2006.

[35] H. Zhang, A. Goel, and R. Govindan. Using the small-world model to improve Freenet performance. *Computer Networks*, 46(4):555–574, 2004.