ON THE STUDY OF THE OPTIMISITC UNCHOKING ALGORITHMS AND INCENTIVE MECHANISMS OF BITTORRENT

ZUHUI MA

A THESIS

IN

THE DEPARTMENT

OF

ELECTRICAL AND COMPUTER ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

SEPTEMBER 2008

Your file Votre référence
ISBN: 978-0-494-45311-7
Our file Notre référence
ISBN: 978-0-494-45311-7

Abstract

On the Study of the Optimisitc Unchoking Algorithms and Incentive Mechanisms of BitTorrent

Zuhui Ma

Optimistic unchoking plays an important role in BitTorrent Peer-to-Peer (P2P)[46, 45, 48, 4] file sharing networks. Peers use optimistic unchoking to find upload bandwidth information about their neighbors. However, free-riders can also take advantage of optimistic unchoking and download from the network without uploading anything. In this thesis, a novel optimistic unchoking algorithm for BitTorrent is proposed. The main purposes of our algorithm are to prevent free-riding and to improve the efficiency of optimistic unchoking. A stochastic model is then proposed to analyze the performance of my algorithm. We also verify the results by simulations.

BitTorrent also have a built-in incentive mechanism called "Tit-for-Tat"[4] to prevent free-riding. Basically, a peer will upload to other peers (default is four) that give it the highest download rate. In this thesis, We will show that by adjusting the upload rate and the number of uploads, a selfish peer can take advantage of the "Tit-for-Tat" [4] to improve its download rate. However, this strategy of the selfish peer is harmful to the whole network. If many peers take the same strategy, the performance of the whole network will be significantly decreased. It is then theoretically proved that the "Tit-for-Tat"[4] is not an optimal incentive mechanism. To solve this problem, We propose a new incentive mechanism for BitTorrent. With this new mechanism, even if all peers are selfish, the performance of the whole network can still be maintained at a very high level.

Acknowledgments

At first, I would like to thank my supervisor Dr. Qiu from my deepest heart. During past two years, His patience and helpful instructions encourage me to further the way of my research. His understanding and warm hearted color my life of education here. He is my role model of hard working, dedication, and confronting challenges, inspiring my interest on research, encouraging me not to give up. Beyond all of mentioned above, he also helped me on details of social life from communication to interview skills.

Subsequently, I would like to thank my officemates in S2-423. Siavash Rahimi, Mojtaba Zokaei Ashtiani who provided many constructive suggestions both on my research and my life, and also enriched the life in the office during these years. Then I would like to thank my friends in Montreal. Liu, Sining and Lei, Ming helped me a lot on my life, and Zhu, Li and Tang, Yang gave me so much advice on my research and career.

Finally, I would like to thank my parents Ma, Hanwen and He, Zhimin who provided me the unconditional support and endless love to encourage me go further in my life.

Contents

List of Figures

Chapter 1

Introduction to Peer-to-Peer Networks

1.1 Overview

With the development of Internet, Peer-to-peer (P2P)[46] networks have attracted a lot of attention in past few years. P2P applications, in which peers serve as both servers and clients, have changed the Internet dramatically. Today, Internet has become a main resource of information and an important media for people to obtain and share information with others. Among the many applications, file sharing is one of the important methods for people, through Internet, to share and obtain information. As evidenced by traffic measurement of ISPs, a large percentage of file sharing is done in P2P fashion.

1.2 P2P Model And Client-Server Model

In P2P model, as we mentioned above, peers have both servers' and clients' responsibilities and capacities, while in the Client-Server model[47], each peer provides either a server's or a client's functionality. Both P2P Model and Client-Server Model are widely used in today's Internet, and also either of models has its own advantages and disadvantages.

Next, we will briefly introduce those two models—Client-Server model and P2P model.

1.2.1 Client-Server model

The Client-Server model provides a convenient way to efficiently interconnect programs that are distributed across different locations in the network. the Client-Server model is wildly used in computer networks. Most of Internet applications, such as email, web access and database access, are based on the Client-Server model[47], and most of Internet's application protocols, such as HTTP, FTP, SMTP[9, 28], etc, are also based on the Client-Server model. Due to the Client-Server architecture, there are several advantages of this model. For example, ease of server maintenance, ease of updating resource, security and service assurance in dedicated servers etc. However, the Client-Server model also has its own disadvantages, such as the bottleneck at the server and the lacking of robustness. For example, as the number of simultaneous client requests to a given server increases, the server can become severely overloaded, which will seriously influence the performance of server. The Client-Server model also lacks the robustness. For example, under Client-Server model, if there is a critical server failure, clients' requests cannot be fulfilled.

1.2.2 P2P Model

Compared to the Client-Server fashion, the P2P model can easily provide the solution to the drawbacks of the Client-Server model. In P2P networks, its total bandwidth actually increases as more nodes are involved, since the P2P network's overall bandwidth can be roughly computed as the sum of the bandwidths of every node in that network. P2P networks normally have very good scalability since the load is usually distributed among many nodes. Even if one or more nodes leave the network, the remaining nodes may still have all the data needed to complete the download, which shows the robustness of P2P networks. The architecture difference of the two models are presented in Fig. 1

In a summary, the P2P model has the following advantages.

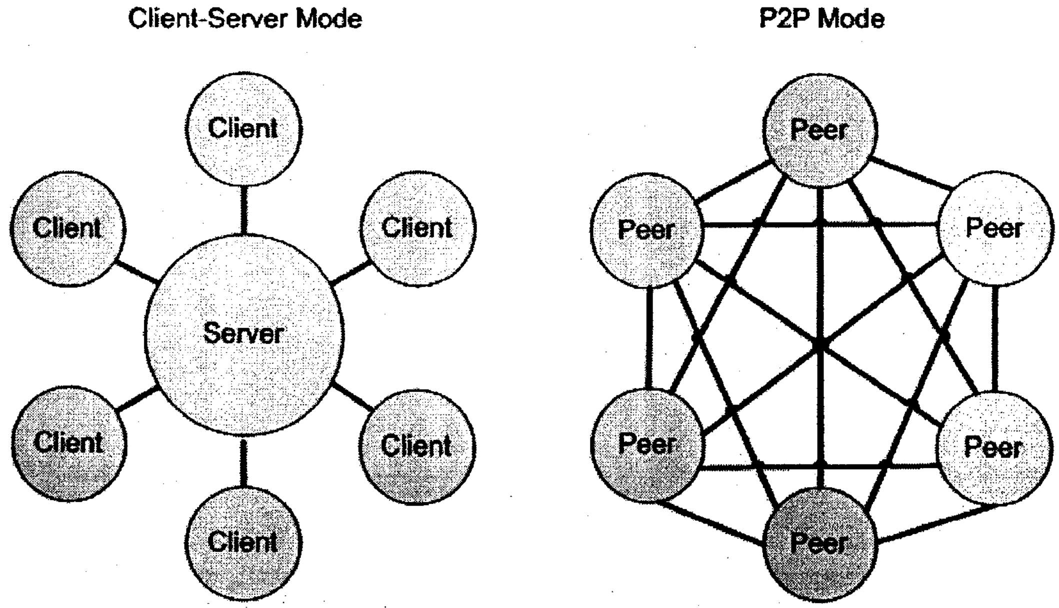

Figure 1: P2P Model and Client-Server Model

Scalability

In the P2P model, peers act as both clients and servers simultaneously. In a well-designed P2P system, more peers generally means better performance since there are more peers to provide service.

Anonymity

Some protocols of P2P network (such as Freenet) can hide the identity of users by communicating through intermediate peers.

Ease of sharing

Peers always select neighbors with best performance to unchoke (upload is called unchoke in P2P networks), and the information of sharing does not need to be flooded through the whole P2P network.

Low cost

In P2P networks, all peers provide resources, such as bandwidth, space of storage, and power. Hence it reduces the cost of the original service provider.

Robustness

Replicating data over multiple peers enables peers to find the data without only counting on a centralized index server.

Due to the attractive architecture of P2P model, many applications have been developed based on P2P model. Examples are file sharing, telephony, media streaming, and discussion forums etc. However, the P2P model is not a substitution for the Client-Server model, because both of the models are needed and valuable in today's network. Instead, the P2P model is an excellent complementary model to the Client-Server model. Some new applications even merge both models together. One example is the Structured overlay Networks Application Platform (Snap)[14] , which significantly improves the performance of traditional Client-Server Model.

1.3 P2P File Sharing

P2P file sharing has been one of the most popular P2P applications. The P2P file sharing refers to uploading and downloading the intended files where the files are stored on individual computers (or peers), over the Internet.

Before P2P file sharing, most of file sharing was done in the fashion of Client-Server. With the help of protocols such as HTTP and FTP, the Client-Server model is widely used over the Internet, because those protocols make the file sharing easy to implement. However, as we described before, if the number of requests from clients increases, the load of server grows quickly and the performance of file sharing deteriorates rapidly. Those shortcomings make the Client-Server model losing its popularity in file sharing, and P2P file sharing, with its excellent scalability, has become immensely popular during recent years.

1.3.1 First generation of P2P file sharing networks

In the first generation of P2P file sharing networks, such as Napster and eDonkey2000 [26, 45, 44, 29], there is a centralized system which is used to control the traffic among the peers in the network. The server only has the information of the shared file and users who upload that shared file to help users who want to download that shared file. For example, if one user sends a search request to the centralized server, then the server will send him/her a list of users who had that file to help him/her to build up connections and to download from them. The first generation P2P file sharing utilizes the bandwidth of peers to reduce the traditional server's burden. However, there are still two weak points of first generation P2P file sharing, which are the bottleneck of the centralized server and out of date information of the shared file. The first generation of P2P file sharing also induced legal problems. Users may require and transmit copyrighted materials with the help of the centralized servers. Hence the operators of central index servers may encounter legal issues.

1.3.2 Second generation of P2P file sharing networks

To solve the bottleneck problem of the centralized sever, the P2P file sharing designers created a network without a central index server, and a new generation—second generation of P2P file sharing appeared. The second generation is improved by means of choosing higher-capacity peers act as index server, and the lower-capacity peers branching off from them. Distributed hash tables (DHTs) are also introduced in the second generation of file sharing systems. By electing various nodes to index certain hashes, which are used to identify files, make up the scalability problem, and increase the efficiency of shared files searching. There are many P2P file sharing applications in the fashion of second generation, such as Gnutella, Kazaa and eMule[44, 46, 45, 29, 3]. However, the higher-capacity peers, which do the similar jobs as central servers, might still encounter legal problems.

1.3.3 Third generation of P2P file sharing networks

To protect the higher-capacity peers and avoid legal troubles, anonymity features, which makes others hard to know who provide shared files and who download and disseminate those popular files, have been introduced in the third generation P2P file sharing. However, the anonymity features are not wildly used in current systems, because the overhead of keeping anonymity features is large. Such large overhead will badly influence the performance of download and efficiency of sharing files.

1.3.4 Fourth generation of P2P file sharing networks

The fourth generation of P2P file sharing emerged for improving the P2P file sharing's performance. In fourth generation of file sharing such as PPlive and PPstream, the exchanging between peers are sending stream instead of files, which makes listening radios and watching television without server possible.

Next, we will introduce one of the most popular P2P file sharing applications, BitTorrent.

1.4 Introduction to BitTorrent

As we discussed before, *Peer-to-Peer* (P2P) applications, in which peers serve as both clients and servers, have changed the Internet dramatically. Compared to traditional client/sever applications (such as *FTP, HTTP*)[9, 28], P2P systems normally have much better scalability. The performance of traditional client/server applications deteriorates rapidly as the number of clients increases, while in a well-designed P2P system, more peers generally means better performance. Among all these P2P applications, file sharing has been one of the most popular. For example, 43% of the Internet traffic is P2P traffic[39]. Among all P2P systems, BitTorrent seems to be the most prevalent one . In particular, more than 50% of all P2P traffic is BitTorrent traffic[33, 21].

BitTorrent is a P2P application whose goal is to facilitate fast downloads of popular files, and also is a method of distributing large amounts of data widely without the original distributor incurring the entire costs of hardware, hosting, and bandwidth resources. When some of popular files are distributed by BitTorrent protocol[48], each peer provides pieces of files and shared with others for reducing the cost and disseminating the burden to individuals.

In the BitTorrent network, there are two categories of peers, namely, downloaders and seeds. A downloader contains some parts (or none) of the file and consumes the resources of the seeds and the other downloaders, while a seed contains the complete file and offers uploading to downloaders.

Next we provide a brief description of how BitTorrent works when a single file is downloaded by many users simultaneously. The shared file is divided into many pieces (the default size of a piece is 256KB). When a peer first joins the network, to facilitate access to information for the peers, it connects to a central server called tracker to get a list of peers (Note that the latest version of BitTorrent supports trackerless torrents, where the centralized tracker is replaced by distributed tracking). The new peer then connects to those peers to request for pieces and those peers become the neighbors of the new peer. Once the new peer obtains at least one pieces, it can start to contribute to the network by uploading pieces. The peer then exchanges pieces with its neighbors until it obtains all the pieces and becomes a seed. Once a peer becomes a seed, it may decide to stay in the network to serve other peers or just leave the network. From the above description, we see that in BitTorrent, a peer is normally a client and a server at the same time. When the network becomes large, there will be more peers to request service, but at the same time, there will also be more peers to contribute to the network. Hence, the performance may not degrade and that is why BitTorrent has good scalability.

Note that the good scalability of BitTorrent is based on the assumption that peers are cooperative and are willing to contribute to the network. However, in reality, a large amount of peers are so-called free-riders. Free-riders are selfish peers that try to download from the network while not contribute (or upload) at all. BitTorrent has a built-in incentive mechanism called "Tit-for-Tat" to encourage peers to upload. A peer in BitTorrent normally chooses a fixed number of other peers (default is four) [48] to upload (also called unchoke in BitTorrent) and those chosen peers are the ones that give the current peer the highest download rates. So basically, if you want to download from others, you should also upload to them as exchange. However, the incentive mechanism also has it own drawbacks. For example, if a new peer with high upload bandwidth joins the network, since it has nothing to upload yet, no other peers will upload to it and hence the high upload bandwidth of the new peer will be wasted. To solve this problem, BitTorrent uses another mechanism called "optimistic unchoking". Besides the normal unchokes we mentioned above, a peer also randomly choose another peer to upload and this is called optimistic unchoking. When a new peer joins the network, it can get its first piece through optimistic unchoking and after that, it can participate in the normal exchange process. Optimistic unchoking is also used to discover peers with high upload bandwidth in the network. By random choosing a peer to upload, it is possible to find a peer with higher upload bandwidth than currently unchoked four peers and hence increase the total download rate.

Besides the protocols we discussed above, there are also other protocols to improve the performance of BitTorrent. One of them is called rarest-first policy (LRF) [4], which is to make sure that the pieces are uniformly distributed among the peers. The LRF increases the availability of a rarest piece of the file. With this policy, a downloader has a preference of the rarest piece among its neighbors, namely, the rarest piece, which has the highest priority to be downloaded first, will be disseminated to other peers. It has been proved that the LRF algorithm can efficiently increase the service capacity and prevent piece problem[1]. Another protocol called endgame mode[4] is used to prevent users who have all but a few pieces of the complete file from waiting too long to finish their download. For example, Sometimes a piece will be requested from a peer with very slow transfer rates. This isn't a problem in the middle of a download, but could potentially delay a download's finish. To keep that from happening, when a peer has only one piece left to download, the peer will send requests for that piece to all its neighbors that have that piece. So the peer doesn't need to wait too long for the last piece.

1.5 Related Work

As one of the most popular applications in the current Internet, BitTorrent has attracted a lot of research interest. Some recent research has emphasized on performance modeling of BitTorrent. In [49], a branching process is used to study the service capacity of BitTorrent-like P2P file sharing in the transient regime and a simple Markovian model is presented to study the steady-state properties. In [11], a simple fluid model is proposed to study the performance and scalability of BitTorrent-like P2P systems. In [31], the authors studied the behavior of peers in BitTorrent and also investigated the file availability and the dying-out process.

The incentive mechanism is another important research topic. In [11], the effect of "Tit-for-Tat" is studied when selfish peers are able to adjust their uploading bandwidth. In [16], an overlay formation game model is used to study the existence of Nash equilibrium and the loss of efficiency when peers can change the number of connections. In [34], it is shown that the "Tit-for-Tat" incentive mechanism is generally not robust to strategic clients. A strategic client can take advantage of "Tit-for-Tat" and hurt the performance of other peers.

1.6 Organization of the Thesis

The thesis is organized as follows:

- In Chapter 2, a novel optimistic unchoking algorithm for BitTorrent is proposed and analyzed. First, We will discuss the motivation of why a new optimistic unchoking algorithm is needed based on the analysis of the current algorithm used in BitTorrent. The details of the new algorithm is then presented and it is mathematically proved that the new algorithm can prevent free-riding behaviors effectively. A stochastic model is then proposed to analyze the performance of the new algorithm. Simulation results are also presented to verify the effectiveness of the proposed new algorithm.

- In Chapter 3, a new incentive mechanism is proposed and analyzed in details. In Section 3.1, We will describe the motivation of the work. In Section 3.2, We propose a strategy for selfish peers and show that the strategy can improve the performance of the individual peer significantly. However, when all peers take the same strategy, simulation results show that the performance of the whole network will be decreased. In Section 3.3, We will show that the built-in incentive mechanism of BitTorrent is not optimal under general assumptions. We then propose a new incentive mechanism for BitTorrent. In Section 3.4, We will use simulation to evaluate the effectiveness of the new incentive mechanism.

- In Chapter 4, We will draw conclusions of the whole thesis and discuss about possible future work.

Chapter 2

Optimistic Unchoking Algorithms for BitTorrent

2.1 Motivation

As we mentioned in Chapter 1, in BitTorrent, with the rate based TFT unchoking scheme, a peer will provide reciprocation to his/her neighbors (default is 4) who provide him/her the highest download rates and to one more, randomly selected neighbor, via a process called optimistic unchoking[21]. In this chapter, we focus on the optimistic unchoking algorithm. As we have described before, the optimistic unchoking not only can explore newly-arrived peers to the network, and also can discover the better performance peers to connect with in the next round. However, optimistic unchoking also introduces new problems. One of them is the free-riding. In [11], it was shown that a free-rider can obtain at least one fifth download rate of a normal peer just from optimistic unchoking. Since all P2P applications are based on the contributions of individual peers, if free-riding becomes serious, the network may not be able to survive. Hence, it is very important to provide a good mechanism to prevent free-riding. In this chapter, we will propose a new optimistic unchoking algorithm which can prevent free-riding much more effectively and at the same time, can improve the performance of normal peers.

2.2 A New Optimistic Unchoking Algorithm

Before we propose the new optimistic unchoking algorithm, let us do a quick review of the purposes of optimistic unchoking and to see why the current algorithm in BitTorrent is not effective. The main purpose of optimistic unchoking is to discover peers with high upload bandwidth and hence to improve the total download rate. The built-in incentive mechanism is not helpful here because the "Tit-for-Tat" only choose from the neighbors that are currently upload to you. If a peer is not uploading to you, the "Tit-for-Tat" will not help you to find what its upload bandwidth is. So, in BitTorrent, besides the four normal unchokes, a peer will also randomly choose a neighbor to unchoke. In its simplest form, a peer may use a round-robin fashion to choose the optimistic unchoking and after several rounds, the peer may be able to discover the upload bandwidth information about its neighbors. The second purpose of optimistic unchoking is to help new joined peers to obtain the first piece quickly as we have discussed in Chapter 1. The current optimistic unchoking algorithm in BitTorrent, however, is not very effective. First, it does not take advantage of the history information. A peer will randomly choose a neighbor as optimistic unchoking even if it knows the neighbor may have a very low upload bandwidth. Secondly, it encourages free-riding. In [11], it was shown that a free-rider can obtain at least one fifth download rate of a normal peer just from optimistic unchoking. Next, we will present a new optimistic unchoking algorithm to solve the mentioned problems.

In our new algorithm, we try to utilize the history information we obtained through past optimistic unchokings. To do so, a peer i need to maintain some information about its neighbors. Let L be the number of its neighbors and $j = 1, \cdots, L$ be the index of a neighbor j. We define the following terms.

N_j The number of times that peer j has been optimistically unchoked.

n_j Among the N_j unchokes, the number of times that peer j responded, i.e., peer j also unchoked (or uploaded to) peer i.

u_j The average upload rate of peer j. Note that if peer j never uploaded to peer i, the information of u_j may not be available.

U_{max} The maximum average upload rate of peer i's neighbors, i.e., $U_{max} = \max u_j$.

Note that in BitTorrent, the default value of L is 20. So the resource required to maintain the history information we proposed here is reasonable. Next, for each neighbor j, we define a gain-value G_j. Basically, G_j is the expected gain or benefit we can obtain if we unchoke peer j in the current time slot (note that here we use a slotted time model for its simplicity). If $n_j > 0$, i.e., peer j has uploaded to intended peers before and we have the upload rate information of j, then

$$G_j = \frac{u_j n_j}{N_j}.$$

If $n_j = 0$, peer j has never uploaded to us before and we don't have any information about its upload rate. In this case, we define

$$G_j = \frac{U_{max}}{N_j + 1},$$

i.e., we use U_{max} as the estimation of the upload rate of peer j. There are two reasons for doing that. First, recall that the main purpose of optimistic unchoking is to discover the upload bandwidth of unknown peers. So, using U_{max} as the estimation will give the unknown peer j a high gain-value and hence a high chance to be unchoked. Secondly, when a new peer join the network, its $N_j = 0$ and hence $G_j = U_{max}$. That means the new peer will be very likely unchoked and this will help the new peer to obtain its first piece as soon as possible.

Once we obtain the gain-value G_j of all neighbors, we choose the peer that give us the highest gain-value to unchoke. In the case of a tie, we randomly choose a peer among those with the highest gain-value.

From the description of the new algorithm, we see that the new algorithm takes advantage of history information and always unchokes the peer with the highest expected gain. While in the original algorithm, a peer chooses the optimistic unchoke randomly and overall it may only obtain the average gain. Hence the new algorithm, on one side, can improve the performance of normal peers. On the other side, the new algorithm can also prevent free-riding. A free-rider will never upload to other peers. So its gain-value will be

$$G_j = \frac{U_{max}}{N_j + 1}.$$

To understand this more clearly, let's assume the system is in a steady state and the highest gain-value converges to a constant $C > 0$. Obviously, when $N_j > \frac{U_{max}}{C} - 1$, the gain-value of the free-rider will always less than C. So the free-rider will only be unchoked a finite number of times. After that, the free-rider will never get any downloading through optimistic unchoking. In other words, a peer can identify a free-rider through a finite number of unchokes and hence effectively prevent free-riding. Note that in the original algorithm, a free-rider can continuously download through optimistic unchoking since peers don't use history information to identify free-riders.

2.3 Stochastic Model

In Section 2.2, we have briefly discussed that our algorithm can improve the downloading performance and prevent free-riding. However, in a real network, peers may dynamically join and leave the system, which will make the history information less accurate and hence may affect the performance of our algorithm. In this section, we propose a stochastic model to analyze this issue.

For the simplicity of analysis, we will first make some assumptions. We assume all normal peers have the same upload rate u. Each peer has a fixed number of neighbors. When a neighbor leaves the system, a new peer will be added to make the number of neighbors a constant L. Note that this is a reasonable assumption because BitTorrent has a mechanism to request for new peers from the tracker if some of its neighbors leave the system. We also assume that a free-rider can be identified through one optimistic unchoke. This is a simplification of the case that a free-rider can be identified by a finite number of unchokes as we have discussed in Section 2.2.

For a given peer, we define the following terms.

X The number of known normal peers in the neighborhood.

Y The number of known free-riders in the neighborhood.

γ The leaving probability of a peer in a given time slot.

p The probability that a newly-arrived peer is a free-rider.

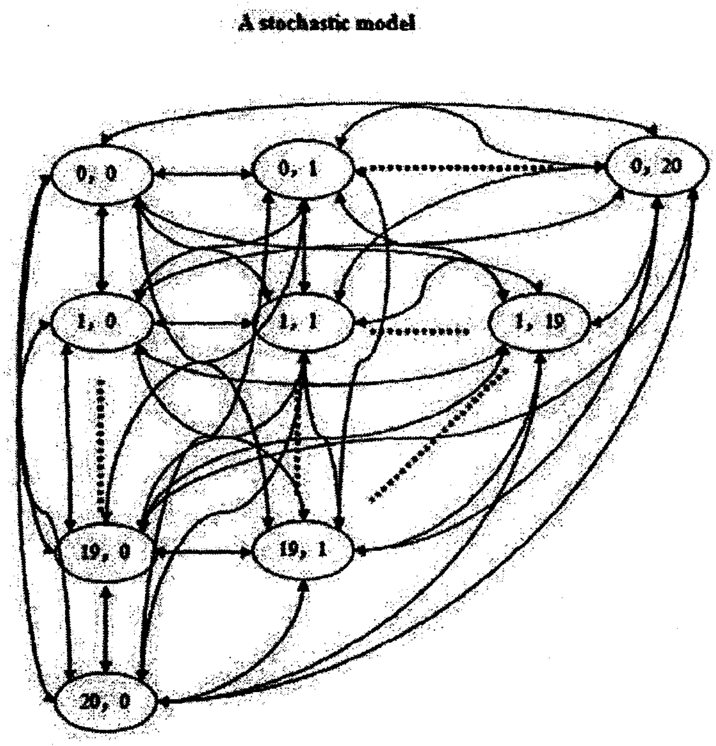

Figure 2: Stochastic Model

Note that since the total number of peers in the neighborhood is L, there will be $L-X-Y$ peers that we don't know if they are normal peers or free-riders. These peers are either newly-arrived or peers that we have never unchoked before.

Now under the assumptions, the random process of (X, Y) becomes a two-dimensional Markov chain. At each time slot, three type of events may occur. First, some known normal peers may leave the system. Let N be the the number of those peers. Then N follow a Binomial distribution with parameters X and γ. Secondly, some known free-riders may also leave the system. Let M be the number of those peers. Then M is also a Binomial random variable with parameters Y and γ. Finally, if $X+Y<L$, that means there are unknown peers in the neighborhood, following our optimistic unchoking algorithm, one of those unknown peers will be chosen to be unchoked and hence will be identified as either a normal peer or a free-rider. Let V be the random variable that indicate that the unknown peer is identified as a free-rider. Then $P\{V=1\}=p$ and $P\{V=0\}=1-p$.

So in a summary, if the Markov chain is in the state (X, Y) and $X+Y<L$, the probability that the state jumps to $(X-n+(1-v), Y-m+v)$ in the next time slot is

$$\begin{aligned} q_{n,m,v} &= \mathrm{P}\{N=n, M=m, V=v\} \\ &= \binom{X}{n}\gamma^n(1-\gamma)^{X-n}\binom{Y}{m}\gamma^m(1-\gamma)^{Y-m}p^v(1-p)^{1-v}. \end{aligned}$$

If $X+Y=L$, the probability that the state jumps to $(X-n, Y-m)$ is

$$\begin{aligned} q_{n,m} &= \mathrm{P}\{N=n, M=m\} \\ &= \binom{X}{n}\gamma^n(1-\gamma)^{X-n}\binom{Y}{m}\gamma^m(1-\gamma)^{Y-m}. \end{aligned}$$

The Markov chain is a complicated two-dimensional chain and it is hard to get a closed-form solution for the steady-state distribution. However, given the jumping probability, we can easily find the numerical solution for the steady-state distribution. Let $\pi(x, y)$ be the steady-state probability that the system is in state (x, y). If we assume that unchoking a normal peer will always get response (i.e., the normal peer will also upload to us), the average download rate a normal peer can obtain will then be

$$D = u\sum_{x=1}^{L}\pi(x, L-x) + u(1-p)(1-\sum_{x=0}^{L}\pi(x, L-x))$$

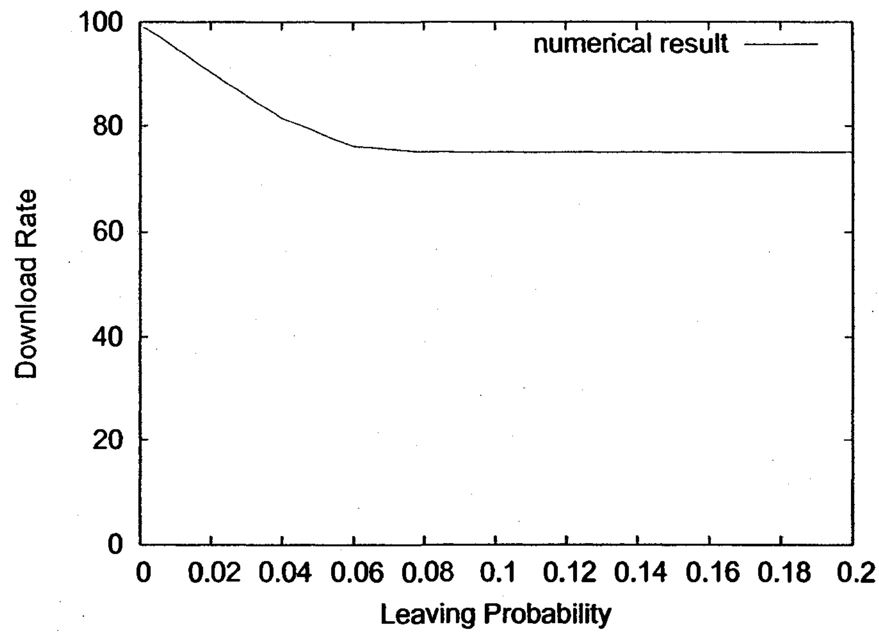

Figure 3: The average download rate of a normal peer

The first term corresponds to the case that all peers are known. So the peer can unchoke a normal peer and hence obtain download rate u. The second term corresponds to the case that there are unknown peers. So a unknown peer will be unchoked and our expected download rate from the unknown peer is $u(1-p)$.

In Fig. 3, we show the numerical result of the average download rate as a function of the peer leaving probability. Here we set $L = 20$, $u = 100$Kbps, and $p = 0.25$. We see that when γ is small, the neighborhood doesn't change frequently and the download rate is close to the up limit 100Kbps. However, when the leaving probability increases, the performance of our algorithm decrease and eventually the download rate becomes 75Kbps, which we can see as the worst case of our algorithm. Note that even the worst case of our algorithm has the same performance as the original optimistic unchoking algorithm, in which, peers are randomly unchoked and hence 75% of unchoked peers are normal ones.

We also simulate a BitTorrent network to verify our stochastic model. We set the same parameters as in the numerical result with $L = 20$, $u = 100$Kbps, and $p = 0.25$. The file of interest is 10M bytes. Every peer in the network will randomly choose $L = 20$ as his/her neighbors when he/she

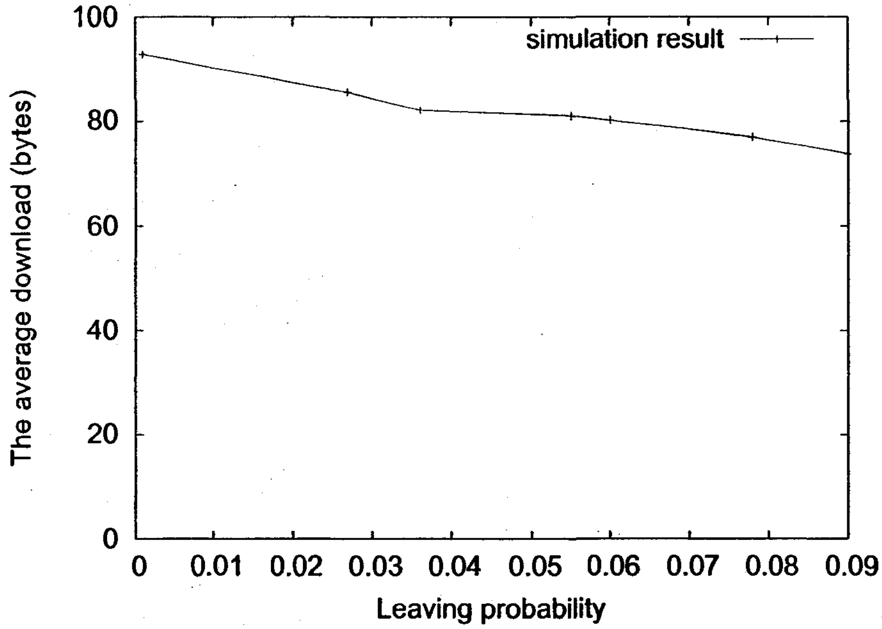

Figure 4: Simulation result of average download via optimistic unchoking

joined network. The newly-arrived peers are followed *Poisson Arrivals* with parameter $\lambda = 5.0$ and peers in the network can leave randomly followed *Exponential Departures* with parameter γ. In our simulation, when the peer finished his/her download, he/she will become a seed automatically for providing the upload to others (if he/she still stay in the network).

In the Fig. 4, we can see that as the leaving probability γ grows, the average download via optimistic unchoking decreases similar to the numerical result. The high leaving probability means that the peer will use optimistic unchoking to test his/her new neighbor more often. Due to newly-arrived peers who are free-riding peers, a non-free-riding peer who is already in the network may lose the chance to get the download via the optimistic unchoking. That is reason why the average download via the optimistic unchoking process goes down as the leaving probability γ goes up. We can also see that in the simulation, the performance decreases more quickly than in the anlysis result. This is because in the stochastic model, we assume that a free-rider can be detected with just one unchoke. While in BitTorrent networks, it may need several unchokes to identify a free-rider.

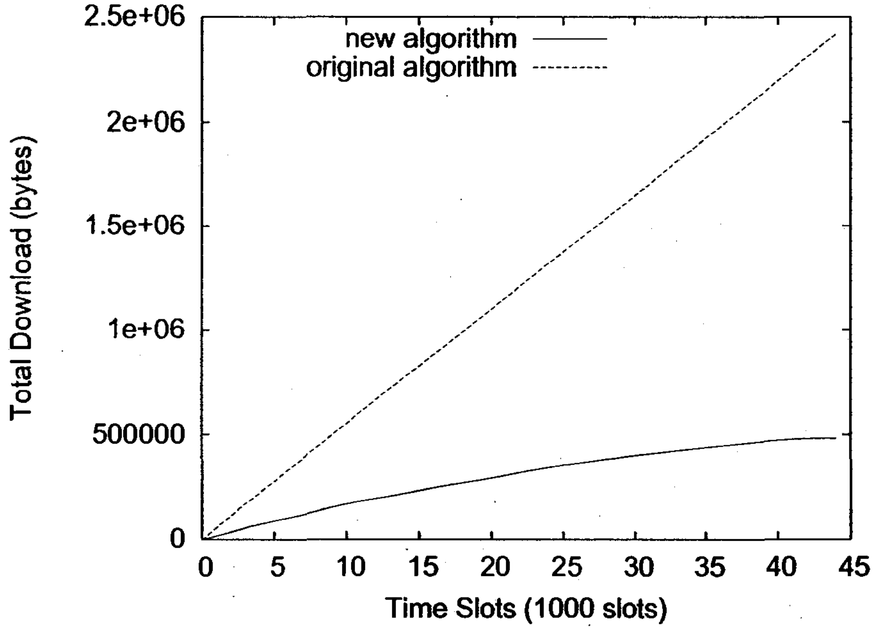

Figure 5: The total download of a free-rider

2.4 Simulation Results

2.4.1 Experiment 1

We have also done extensive simulations to evaluate the proposed algorithm. In experiment 1, there are $L = 30$ peers in a neighborhood. All peers have the same upload bandwidth 100KB per time slot and among these peers, 50% are free-riders. In Fig. 5, we show the total download of a free-rider as a function of time. We can see that with the original algorithm, a free-rider can continuously obtain data from the network through optimistic unchoking. While with the proposed algorithm, the data a free-rider can download from the network is significantly reduced and hence our algorithm can effectively prevent free-riding. In Fig. 6, the total download of a normal peer through optimistic unchoking is shown as a function of time. We see that with the new algorithm, the download a normal peer can get from optimistic unchoking is almost doubled. So our algorithm can also improve the performance of normal peers significantly.

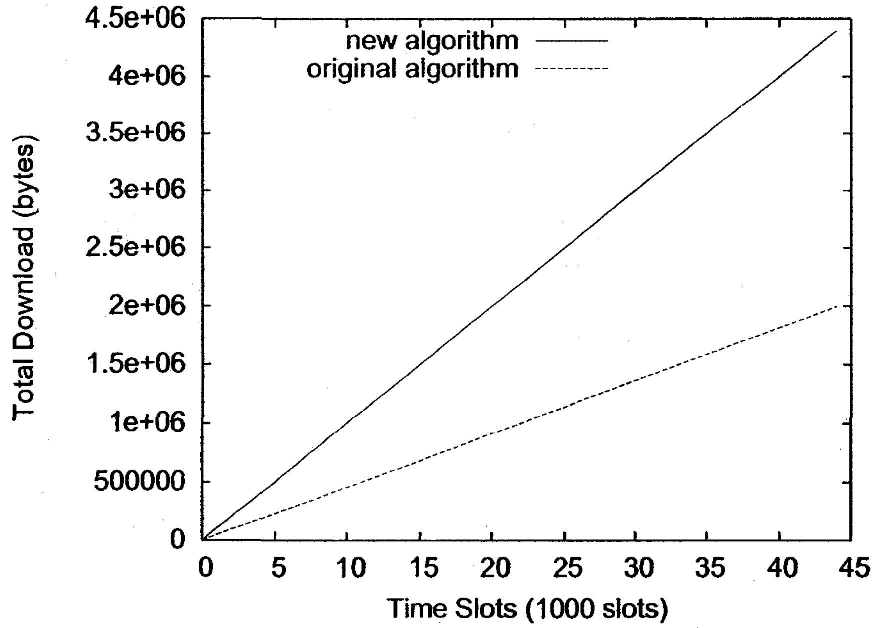

Figure 6: The total download of a normal peer

2.4.2 Experiment 2

In Experiment 2, the maximum number of peers in the network is 500 peers and among the peers 25% is free-riders, and also there are two categories peers with different upload rate, one is 20Kbps, the other is 100Kbps. At begin, the network is empty, peers will arrive according the *Poisson Arrival* with parameter $\lambda = 5$ for downloading the one popular file which is 10M, the peers also can leave according to the *Exponential Departures* with parameter $\gamma = 0.001$. Once the peer finishes his/her download, he/she will become a seed automatically if he/she is still living in the network. In Fig 7. we show the average download of free-riding peers in different algorithms as a function of time. We can see that the free-riders' downloads are effectively prevented by our new algorithm: the average download of free-rider in our algorithm is just a quarter of the average download of free-rider in traditional algorithm. With the traditional algorithm, a free-rider can still keep his/her average download by getting the download from other non-free-riders continuously. In Fig 8 9. we group two capacities peers separated for presenting the performance of optimistic unchoking respectively. The peer's average download rate via optimistic unchoking who has 20Kbps upload rate is shown

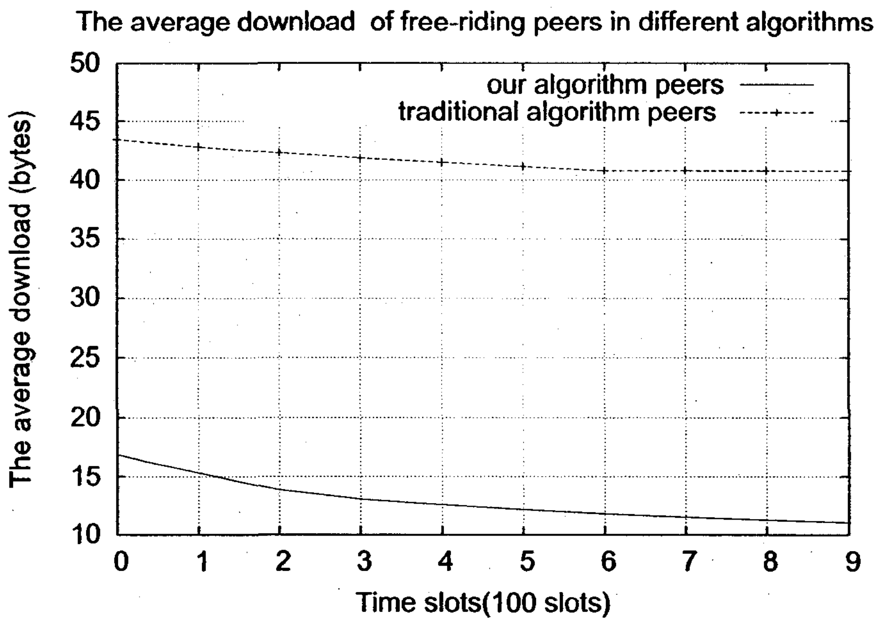

Figure 7: The average download of free-riding peers in different algorithms

as the function of the time, and also the peers' average download rate via optimistic unchoking who has 100Kbps upload rate is demonstrated as the function of the time. In Fig 8 9, we can see that the peers using history to choose optimistic unchoking in our algorithm will effectively improve their average download through optimistic unchoking process, because the peers will significantly reduce the chance to unchoke free-rider peers during their optimistic unchoking process. Again, we see that our algorithm not only can prevent the free-riding peers but also improve the normal peers' performance significantly.

2.5 Conclusion

In this chapter, we proposed a novel optimistic unchoking algorithm for BitTorrent. The basic idea is to take advantage of peer information obtained from past experience and try to unchoke the peer with the highest expected gain. Our theoretical analysis and simulation results show that the new algorithm can significantly improve the performance of normal peers and at the same time, effectively prevent free-riding.

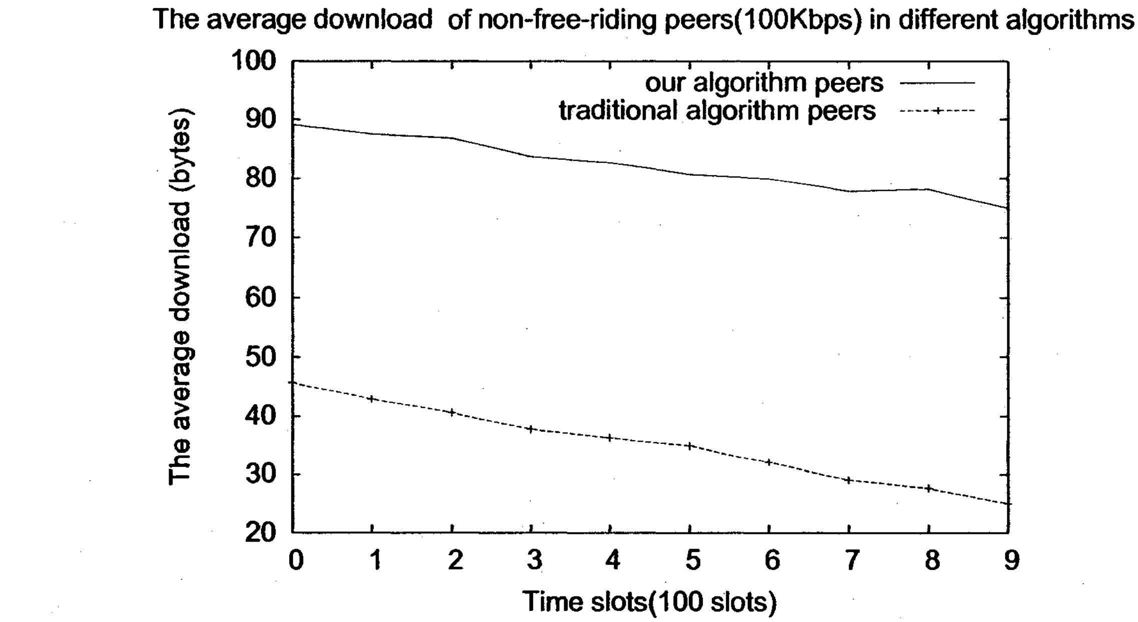

Figure 8: The average download of non-free-riding peers(100Kbps) via optimistic unchoking in different algorithms

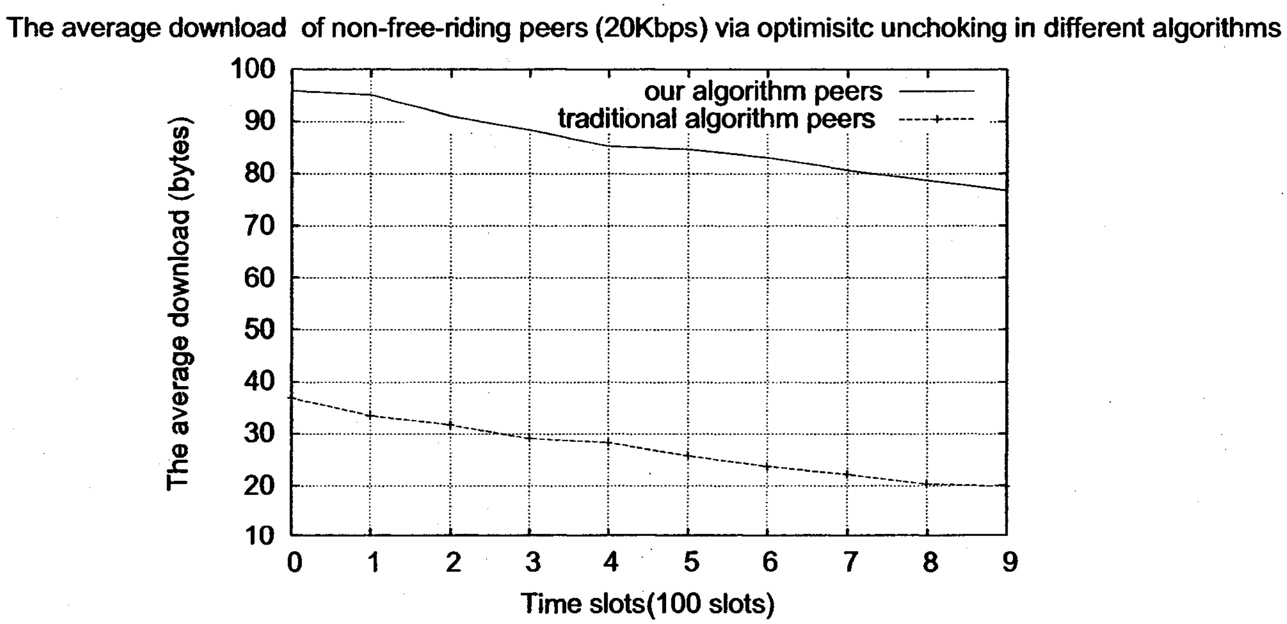

Figure 9: The average download of non-free-riding peers(20Kbps) via optimistic unchoking in different algorithms

One of the advantages of our algorithm is that it can be deployed progressively. A peer can use our algorithm to improve its performance no matter other peers use the same optimistic unchoking algorithm or not. In addition, our algorithm can work with any incentive mechanisms proposed for BitTorrent and hence it can be easily integrated into BitTorrent networks.

Chapter 3

Incentive Mechanisms of BitTorrent

3.1 Motivation

As we have discussed in Chapter 1, BitTorrent have a built-in incentive mechanism called "Tit-for-Tat" to prevent free-riding. Basically, a peer will upload to other peers (default is four) that give it the highest download rate. In [11], it has been shown that a selfish peer can adjust its upload bandwidth to take advantage of the TFT. In [16], it is shown that peers can also adjusting the number of uploads to improve its download rate. From the above two papers, we can see that selfish peers can have the following two strategies to improve their performance of download.

- Maximize the utilization of each upload. For a given peer, we can see that the download rate it obtained from other peers as the gain while the upload as the cost. A selfish peer will try to maximize its gain while at the same time minimize its cost. Note that with the TFT mechanism of BitTorrent, a peer may be able to maintain the same download rate as long as its upload rate is greater than some threshold [11]. Hence, a selfish peer will prefer to set its upload rate just slightly greater than the threshold.

- Adjust the number of uploads. In BitTorrent, the default number of normal uploads is four. A selfish peer, however, can change the number of uploads to gain higher download rate. There is a tradeoff though. If the number of uploads is too large, each upload rate will be small and hence the total download rate obtained may decrease. A selfish peer will try to find an optimal number of uploads to maximize its download rate.

In the next section, we will first propose a strategy for selfish peers to improve performance by means of altering the upload bandwidth and number of uploads at the same time. We will show that a selfish peer can significantly improve his/her own download performance by using our strategy. We will also show that if all peers become strategic, the performance of the whole network (based on our simulations) will be undermined. We will also prove that the BitTorrent built-in incentive mechanism is not optimal when considering the performance of the whole network. Finally, we will propose our novel incentive mechanism for the BitTorrent which will maximize the performance of the whole network even if all the peers are selfish.

3.2 Study of Selfish Peers

In [11], it has been shown that with the Tit-for-Tat, selfish peers are able to adjust the uploading bandwidth for getting more benefits. It has also been proved in [11] that under certain conditions, the whole network will converge to a Nash Equilibrium in which, each peer will set its upload bandwidth to be the physical upload bandwidth. Next we will provide another example which shows that changing the connection numbers is another proposing method to improve peer's performance of download rate.

An Illustrative Example.

Let us take 10 peers divided into two groups as an example. Five peers have physical upload bandwidth $C_1 = 3$ and the other five have bandwidth $C_2 = 2$. Assume that the default number of connections is $n = 3$. According to [11], peers would fully use their upload bandwidth. A possible overlay network is shown in Figure 10, where big circles and small circle represent high-bandwidth

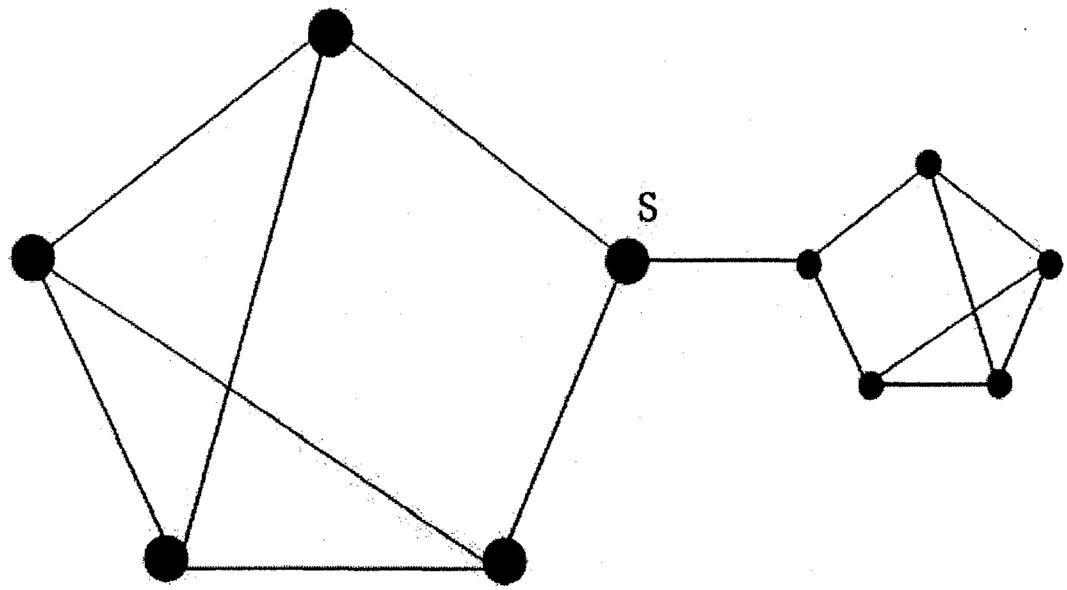

Figure 10: The peers connection structure if peers only adjusting upload bandwidth

and low-bandwidth peers respectively. Note that in this example, the peers do not receive the same download rate, even if they belong to the same group. Four high-bandwidth peers receive a download rate of 3 ($= 3 * \frac{C_1}{3} = 3$), while the peer connecting the two groups (peer S in the figure) receives only 8/3 ($= 2 * \frac{C_1}{3} + \frac{C_2}{3}$). Similarly four low bandwidth peers receive rate of 2, while the other receives 7/3. According to [14] the formed overlay network is stable in the sense that no peer wants to change a link (or reduces its uploading rate) if peers can only adjust their upload bandwidth.

Let us now remove the constraint on the number of connections. For example, peer S decides to increase its number of connections to 5. If all other peers keep $n = 3$, the new equilibrium is presented in Figure 11. Note that peer S improves its performance, because its download rate increases from 8/3 to 10/3 ($= 5 * \frac{C_2}{3}$). This example shows us that peers can benefit by changing their numbers of connections.

Next, we will use simulation to study the effect of selfish peers that adjusting their upload bandwidth and the number of uploads. We will first show that if only a single peer take this strategy, its download rate can be significantly improved. However, if all peers become selfish to dynamically changing their upload bandwidth and the number of uploads, the performance of the whole network will get worse.

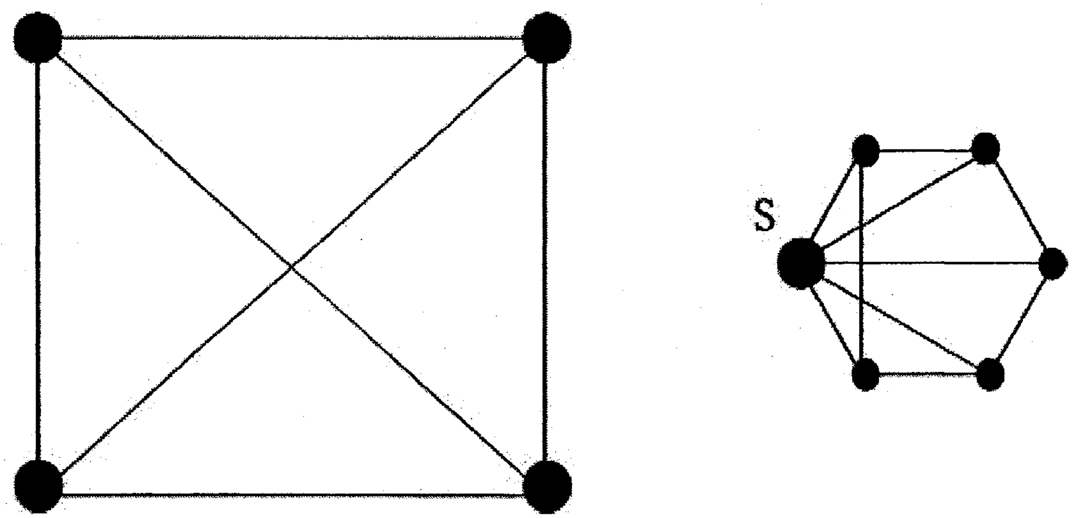

Figure 11: The peers connection structure by changing connection numbers

3.2.1 Simulation results

Since we are interested in the incentive mechanism here, in our simulation, we do not consider the optimistic unchoke, but only the normal unchokes. Note that the effect of optimistic unchoking has been studied in Chapter 2.

3.2.2 Experiment 1

In Experiment 1, there are $L = 20$ peers in a neighborhood. All peers have the same physical upload bandwidth of 400Kbps. There is a single peer that is selfish and use the strategy we described before. All other peers are cooperative and do not adjust their upload bandwidth and the number of uploads. In Fig. 12, we show the total download of a single peer as a function of time. We can see that the total download of a strategy peer is nearly double that of a traditional one. From this simulation, we see that if only one peer is strategic, i.e., dynamically changing connection numbers and upload bandwidth and other peers are implemented in the traditional way, the download rate of the strategic peer can be greatly improved.

However, if all peers become strategic, the whole BitTorrent network will have worse performance. We will show this in the next simulation.

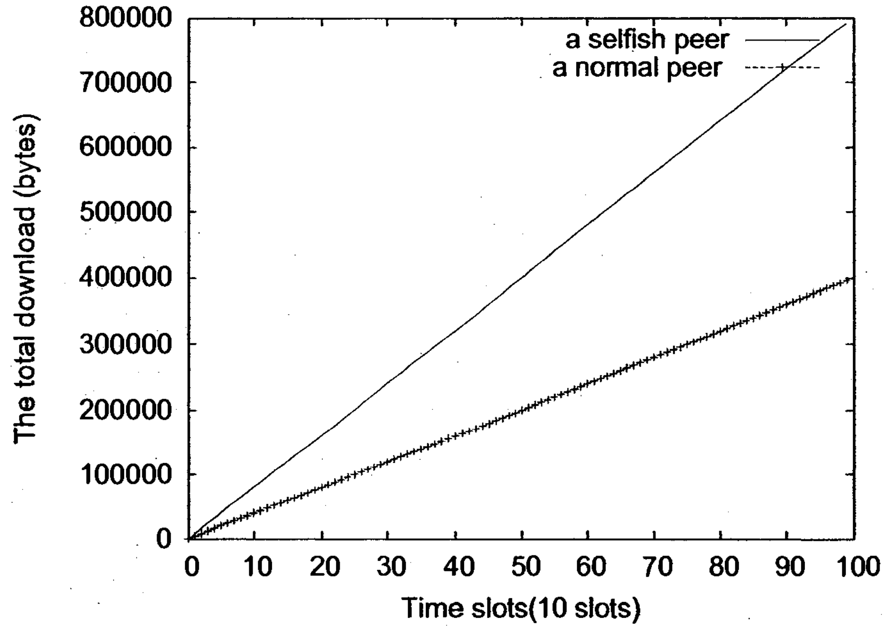

Figure 12: Comparison the total download between a selfish peer and a normal peer

3.2.3 Experiment 2

In Experiment 2, we still have $L = 20$ peers in a neighborhood. All peers have upload bandwidth from 400Kbps to 800Kbps randomly. We compare the average download of two peers in two different settings as a function of time. In the first setting, all peers are cooperative and follow the original algorithm in BitTorrent. In the second setting, all peers are selfish and use the strategy we have described before. From Fig. 13, we see that when all peers are selfish, the performance of the network is in fact worse than when peers use the traditional method. Hence, selfish and strategic peers are inadvisable from the network's point of view.

From the above two simulations, we see that the "Tit-for-Tat" incentive mechanism of BitTorrent in general is not a good incentive mechanism. From a game theory point of view, a well-designed incentive mechanism should maximize the total download rate even when all peers are selfish. While from the simulations, we see that the "Tit-for-Tat" does not have this feature. In the next section, we will explain why the "Tit-for-Tat" is not an optimal mechanism theoretically and then propose a new incentive mechanism for BitTorrent.

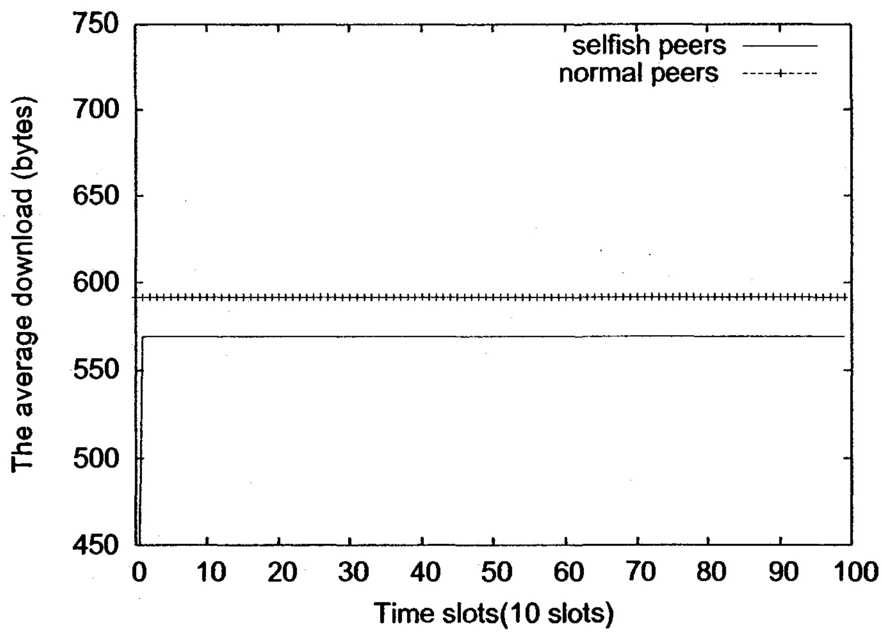

Figure 13: Comparison average download rate between selfish peers and normal peers

3.3 A Novel Incentive Mechanism

Before we propose the new incentive mechanism for BitTorrent, we will briefly explain why the build-in mechanism of BitTorrent is not desirable. For the simplicity of analysis, we assume that each peer in the network has information of its neighbors (such as the upload rate, the connection number etc). We also assume that peers are selfish, namely, each peer will try to maximize its own download rate and minimize its upload bandwidth simultaneously.

We define an incentive mechanism to be optimal if with this mechanism, all peers (either selfish or cooperative ones) will set their upload bandwidth to be the physical upload bandwidth. Note that from the network's point of view, the total download rate is the performance measure that we are interested in. In a network, however, the total download is equal to the total upload. Hence, as long as we maximize the total upload, we will automatically maximize the total download. If an incentive mechanism can encourage peers to set their upload bandwidth to be the physical upload bandwidth, then it will maximize the total upload and hence it is optimal. Next, we will show that the "Tit-for-Tat" doesn't satisfy this condition.

In BitTorrent, all upload and download are done through TCP. For the simplicity of analysis, we assume that if a peer i has upload bandwidth u_i and it unchokes n_i other peers, then the upload rate of each unchoke is $\frac{u_i}{n_i}$, i.e., the upload bandwidth is evenly distributed to the n_i connections. We suppose that there are N peers in the network with their physical upload bandwidth to be $U_1, U_2, U_3, \ldots, U_N$. For a given peer i, we assume that its upload bandwidth is original set to be the physical upload bandwidth U_i and it is unchoked by k peers $j_1, j_2, \cdots, j_k$. Then the total download rate of peer i is

$$D_i = \sum_{l=1}^{k} d_{j_l},$$

where d_{j_l} is the download rate i gets from its neighbor j_l. Now, let's have a look at peer j_l. Assume peer j_l unchokes m peers, then according to the "Tit-for-Tat", peer i must be in the top m peers that give peer j_l high download rate. An important fact here is that as long as peer i is in the top m, it will be unchoked by peer j_l. If peer i is a selfish peer, it can reduce its upload bandwidth as long as it is still in the top m. Peer i can do the same thing for all j_l where $l = 1, 2, \cdots, k$. In a summary, in general, peer i can still maintain its total download rate D_i while at the same time reducing its upload bandwidth. So, with "Tit-for-Tat", a selfish peer in general will not set its upload bandwidth to be the physical upload bandwidth. Instead, it will be strictly less than the physical upload bandwidth in most cases. In this sense, we say that the build-in incentive mechanism "Tit-for-Tat" is not optimal because it will reduce the total download rate of the network. This also explain why in our simulation, when all peers are selfish, the total performance becomes worse.

Next, we will explain this in more details from a peer's point of view. We can find the answers from the protocols of BitTorrent and TCP. In TCP networks, as we assumed before, the total upload bandwidth is evenly distributed to each connections. For example, one peer whose upload capacity is 400Kbps, will evenly assign its upload capacity to each connection (100 Kbps per connection if there are four uploads). In BitTorrent protocol, each peer will unchoke other peers who provide higher download among its neighbors. So the unchoked peer does not need to be the neighbor that give the highest download rate as long as it is in the top 4 (default is 4 connections). In other words, to keep to be unchoked, a peer can reduce its upload rate as long as it is slightly higher than the

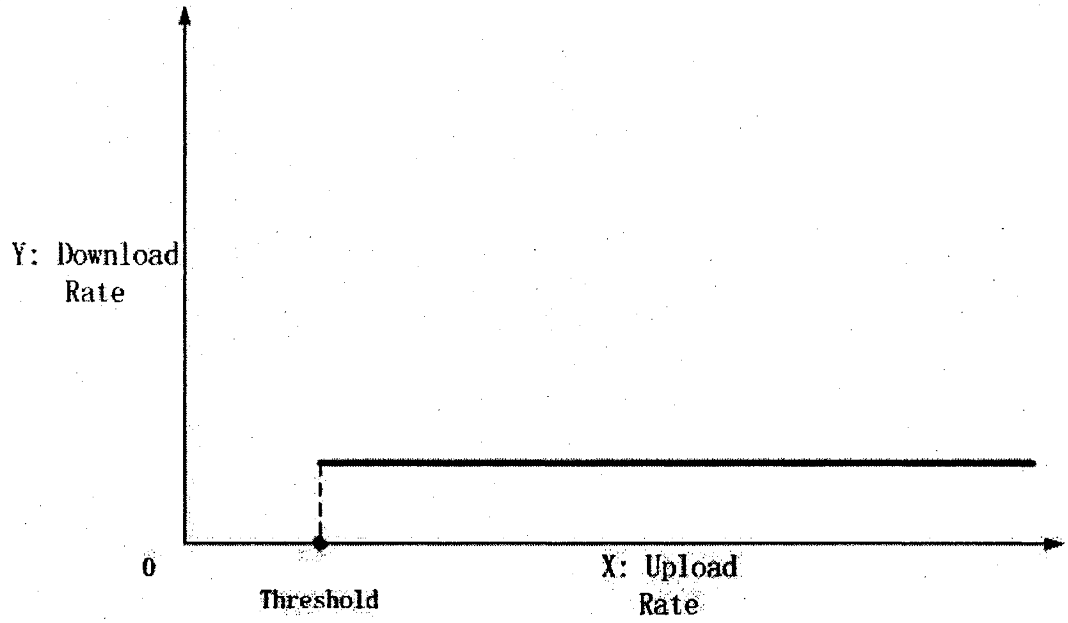

Figure 14: The relation between upload and download rates

fourth one. In other word, as long as a peer's upload rate is greater than some threshold, it can maintain its download from its neighbors. We show this more clearly in Fig. 14

In the figure, we show the peer's download rate as a function of its upload rate between two connected peers (Peer i and Peer j). We can see that when peer i's upload rate exceeds the threshold of peer j, peer i will be unchoked by peer j and hence obtain downloads from peer j. But once it is greater than the threshold, increasing the upload rate further will not help peer i anymore since it will not gain any increase in the download rate. In this situation, we say that peer i has no incentive to raise its upload rate as long as the upload rate is greater than the threshold. Motivated by this fact, we design a new incentive mechanism for BitTorrent which will encourage peers to raise their upload rate to gain more download. The desired relation between the download and upload rate is shown in Fig. 15. Unlike what we have in Fig. 14, here peer i's download rate will grow as the increase of peer i's upload rate. Hence, peers will have incentive to increase their upload rates.

Let's consider the connection between two Peer i and Peer j. We define the following terms.

X The rate that peer i upload to peer j.

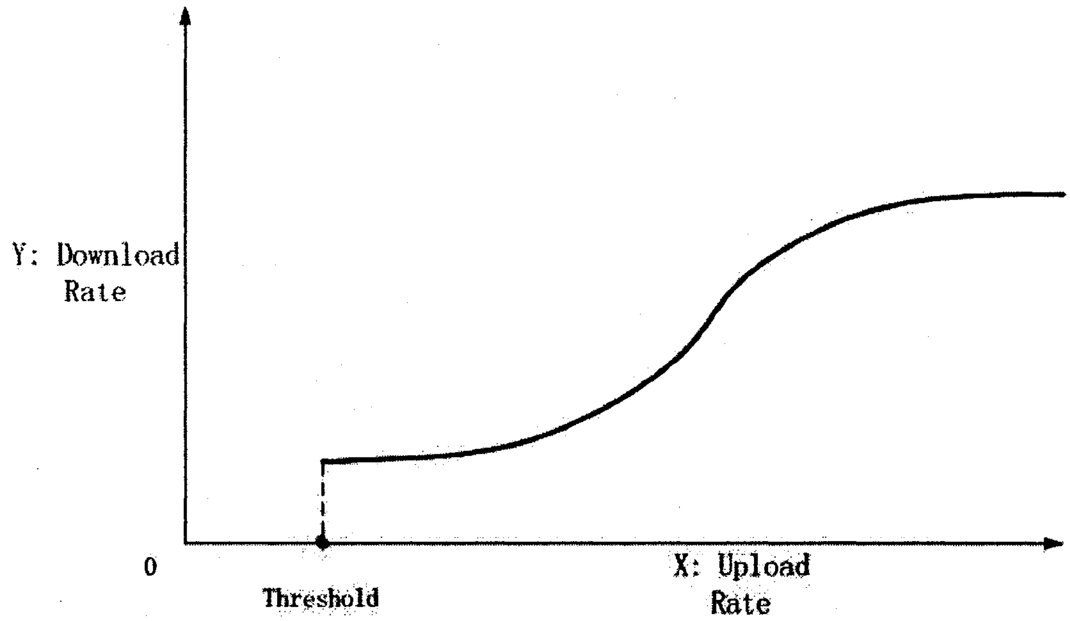

Figure 15: The desired relation between upload and download rates

Y The download rate that peer i obtained from peer j.

TT The threshold of peer j, i.e., if the upload rate of peer i is greater than TT, the peer i will be unchoked.

U_i The physical upload bandwidth of peer i.

Up_i The final allocated upload bandwidth of peer i.

In this thesis, we use following formula to represent the desired relation between Y and X.

$$Y = \begin{cases} \arctan(X - T) + M & \text{if } X \geq TT \\ 0 & \text{if } X < TT \end{cases}$$

where T and M are two constants. Note that this is not the only function that satisfy the desired relation. Other functions may also be used.

Next, we will provide the details of our incentive mechanism. Supposed that there are N peers in the BitTorrent network. For a given peer i, assume that there are K peers that upload to i. In other words, peer i can download from K peers in the network. Without loss of generality, let the download rate that peer i obtained from the K peers be

$$X_1 \geq X_2 \geq X_3 \ldots \geq X_{K-1} \geq X_k$$

Subsequently, peer i will use the above-mentioned equation to calculate the upload rate which will assign to those K peers. However, because peer i's upload bandwidth is limited, peer i will reallocate its upload rate according to its physical upload bandwidth. Let $Y_1, Y_2, Y_3 \ldots, Y_{K-1}, Y_K$ be the calculated upload rates from the above-mentioned equation. Then peer i will reallocate its upload rate as follows:

$$Up_1 = \frac{Y_1}{\sum_{i=1}^{K} Y_i} U_i.$$

$$Up_2 = \frac{Y_2}{\sum_{i=1}^{K} Y_i} U_i.$$

$$\ldots$$

$$Up_{K-1} = \frac{Y_{K-1}}{\sum_{i=1}^{K} Y_i} U_i.$$

$$Up_K = \frac{Y_K}{\sum_{i=1}^{K} Y_i} U_i.$$

The final reallocated upload rates should satisfy:

$$Up_1 \geq Up_2 \geq \ldots \geq Up_{K-1} \geq Up_K$$

Basically, the higher the upload rate, the higher the download rate a peer can get from its neighbors. Note that the parameters such as M, T, TT can be adjusted for different peers. For example, the threshold of TT can be changed to make the connection number flexible.

3.4 Simulation Results

In this section, we will present the simulation results to evaluate our new incentive mechanism.

3.4.1 Experiment 1

In Experiment 1, we set $L = 20$ peers in a neighborhood. All peers have upload bandwidth from 400Kbps to 800Kbps randomly. According to our new incentive mechanism, we set the threshold

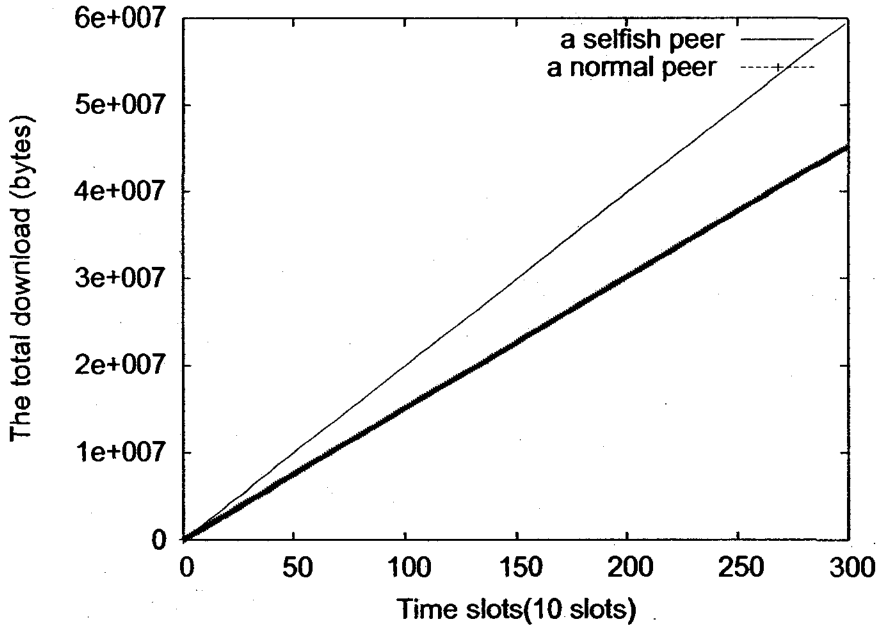

Figure 16: Comparison of download rate between a strategic peer and a ordinary peer under the new incentive mechanism

$TT = \frac{1}{4}$ physical upload bandwidth and the constant of $M = \frac{1}{4}$ physical upload bandwidth for each peers in the network. We do not consider the download part by optimistic unchoking, and free-riding peers in neighborhood, same as the previous experiment assumption. We show the total download of two peers under our new incentive mechanisms as a function of time. One peer is cooperative and the other peer is selfish by taking the strategy we proposed before. In Fig. 16, We can see that under our incentive mechanism, by means of making peers more strategic, peers can get total download from their neighbors $\frac{1}{4}$ more than that under the traditional fashion. This result is similar to what we get with the "Tit-for-Tat".

3.4.2 Experiment 2

In this simulation, There are $L = 20$ peers in a neighborhood. All peers' physical upload bandwidths are from 400Kbps to 800 Kbps randomly distributed. We also set the threshold $TT = \frac{1}{4}$ physical upload bandwidth and the constant of $M = \frac{1}{4}$ physical upload bandwidth for each peer in the BitTorrent network. As the previous experiment, we do not consider optimistic unchoking download

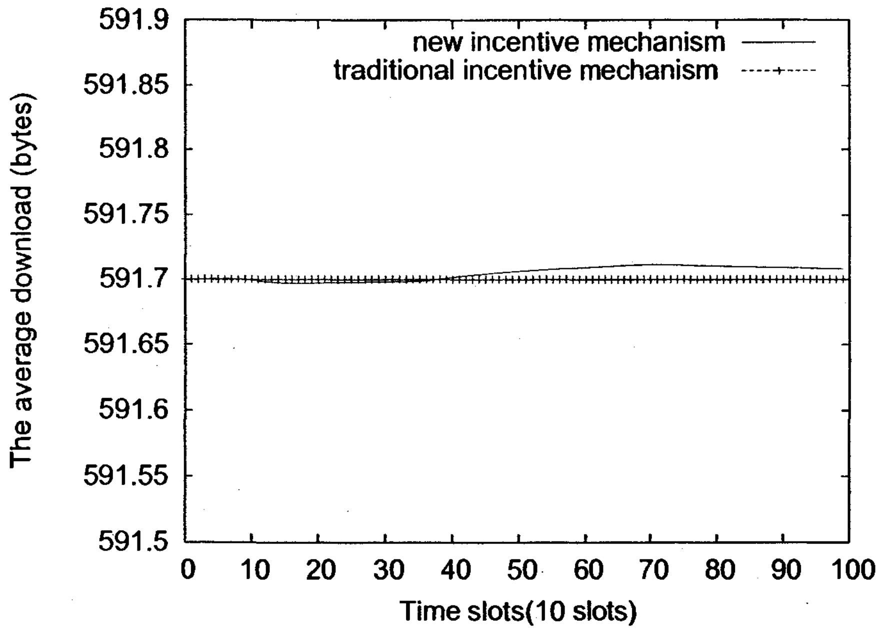

Figure 17: Comparison of the average download rate of all peers under different incentive mechanism

part, and free-riding peers in this simulation. We show the average download rate of all peers in the network as the function of the time. In Fig. 17, we compare the average download rate under two different settings. In the first setting, all peers are cooperative and the incentive mechanism is "Tit-for-Tat". In the second setting, all peers are selfish and the incentive mechanism is what we proposed. We can see that with our proposed incentive mechanism, even when all peers are selfish, the performance of the total network is still maintained at a high level and is similar to that of when all peers are cooperative with the original mechanism.

3.5 Conclusion

In this chapter, we have proposed a novel incentive mechanism for BitTorrent. The basic idea is to encourage peers contribution by reallocating the upload bandwidth according to the download that peer contributed. Our simulation results show that the new incentive mechanism can significantly improve the performance of single peer and at the same time, do not undermine the performance of the whole network. Due to the increasing peers' contribution, our incentive mechanism will discourage the free riders behaviors in the real BitTorrent. We also proved that the BitTorrent's incentive mechanism can not achieve the optimal performance of the network, and BitTorrent robustness which is wildly believed is not truthful in the real BitTorrent networks.

Chapter 4

Conclusion and Future Works

4.1 Conclusion

In chapter 2, we proposed a novel optimistic unchoking algorithm for BitTorrent. By means of evaluating the *gain-value*, which utilizes the history of the peers' established connections, to make decision-making process of optimistic unchoking. Our theoretical analysis and simulation results show that our algorithm of optimistic unchoking, not only significantly improves non-free-riders performance, but also effectively prevents free-riders behaviors. According to the new algorithm of optimistic unchoking, peers will become more strategic than the traditional algorithm in the BitTorrent networks. Our contributions in the chapter 2 are:

- Our new algorithm of optimistic unchoking, considers the efficiency of exploring the newly-arrived peers, at same time, effectively using optimistic unchoking to improve performance of non-free-riders and significantly restricts free-riders.
- Our algorithm of optimistic unchoking can be used to improve its performance regardless of other peers use the same optimistic unchoking algorithm or not. In addition, our algorithm can be employed with any incentive mechanisms which BitTorrent used and can be easily integrated into BitTorrent. It is especially useful for the future BitTorrent designers.

In chapter 3, intrigued by [11, 16], we have proposed a new incentive mechanism to encourage peers to upload. We first show that the built-in incentive mechanism of BitTorrent is not optimal. Selfish peers can take advantage of the incentive mechanism while at the same time, hurt the performance of the whole network. In our incentive mechanism, peers make more contribution to increase their own download rate, in other words, our new incentive mechanism alleviates free-riders problem indirectly, while at the same time, does not undermine the performance of the whole network.

Our contributions in the chapter 3 are:

- We propose a novel incentive mechanism to make peers willingly increase their uploads, and ease the free-riders problem.
- We show that the built-in incentive mechanism of BitTorrent is not optimal.

4.2 Future Work

Although in Chapter 2 the stochastic model can capture all the important features of BitTorrent optimistic unchoking algorithm, it is still relatively simple. For example, in our stochastic model, we assume that all peers have the same upload bandwidth. However, in real networks, peers may have various upload bandwidths and to make optimistic unchoking decision more complicated to analyze, while the available bandwidth is not stable with time. One of the future works is to take the peer heterogeneous upload bandwidths into account in our model.

In chapter 3, we propose a new incentive mechanism to encourage peers to improve their contribution. However, as we mentioned before, many functions may be used to calculate the upload rate. So another future work we need to consider is that trying to find the optimal function between download and upload. Another potential problem in our proposed incentive mechanism is that peers may have different download bandwidth. For example, some of peers have a high capacity of download, and others have a lower one. If one peer with high capacity save its bandwidth to connect with the lower one for increasing its own download, the lower peer may not be able to consume that download from the peer with high capacity. This is another part of our future work.

Bibliography

[1] A.Legout, G.U.Keller, and P.Michiardi. Rarest first and choke algorithm are enough. IMC, Oct.2006.

[2] David Barka. An introduction to peer-to-peer computing. February, 2000.

[3] BBCNews. Kazaa site becomes legal service, 2006. http://news.bbc.co.uk/2/hi/science/nature/5220406.stm.

[4] B.Cohen. Incentives build robustness in bittorrent. IPTPS, 2003.

[5] Berlios. The fasttrack protocol, 2004. http://cvs.berlios.de/cgi-bin/viewcvs.cgi/gift-fasttrack/giFTFastTrack/PROTOCOLrev=HEAD content-type=text/vnd.viewcvs-markup.

[6] A. Bharambe, C. Herley, and V. Padmanabhan. Analyzing and improving bittorrent performance. IEEE INFOCOM, 2006.

[7] A. Bharambe, C. Herley, and V. Padmanabhan. Microsoft research simulator for the bittorrent protocol, February 2007. http://www.research.microsoft.com/projects/btsim.

[8] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. ACM SIGCOMM, 2002.

[9] John Davidson. *An Introduction to TCP/IP.* Springer-Verlag, December 11, 1989.

[10] G. de Veciana and X. Yang. Fairness, incentives and performance in peer-to-peer networks. Monticello, IL, Oct. 2003. Forty-first Annual Allerton Conference on Communication, Control and Computing.

[11] D.Qiu and R.Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer network. SIGCOMM, Sep.2004.

[12] Can Erten and Richard MacManus. Introduction and real world application, March 29, 2007. http://www.readwriteweb.com/archives/p2p introduction real world applications.php.

[13] B. Fan, D.-M. Chiu, and J. Lui. Stochastic differential equation approach to model bittorrent-like file sharing systems. 14th IEEE International Workshop on Quality of Service, 2006.

[14] Carles Pairot Gavalda, Pedro Garcia Lopez, and Ruben Mondejar Andreu. Deploying wide-area applications is a snap. IEEE Internet Computing, 2007.

[15] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang. Measurements, analysis, and modeling of bittorrent-like systems. pages 35–48. 5th ACM SIGCOMM conference on Internet Measurement IMC, 2005.

[16] H.Zhang, G.Neglia, D.Towsley, and G.L.Presti. On unstructured file sharing networks. INFOCOM, 2007.

[17] M. Jackson and A. Wolinsky. A strategic model of economic and social networks. page vol. 29(3). Journal of Economic Theory, 2004.

[18] R. Johari and J. Tsitsiklis. Efficiency loss in a network resource allocation game. pages vol. 71, No. 1, 44–47. Mathematics of Operations Research, 1996.

[19] S. Jun and M. Ahamad. Incentives in bittorrent induce free riding. P2PECON, 2005.

[20] W.-C. Liao, F. Papadopoulos, and K. Psounis. An efficient algorithm for resource sharing in peer-to-peer networks. IFIP Networking, 2006.

[21] Wei-Cherng Liao, Fragkiskos Papadopoulos, and Konstantinos Psounis. Performance analysis of bittorrent-like systems with heterogeneous users. pages Volume 64 Issue 9–12. Elsevier Science Publishers B. V, October 2007.

[22] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang. Exploiting bittorrent for fun (but not profit). International Workshop on Peer-to-Peer Systems, IPTPS, 2006.

[23] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free riding in bittorrent is cheap. HotNets, 2006.

[24] Zuhui Ma and Dongyu Qiu. A new incentive mechanism for bittorrent. 2008.

[25] Zuhui Ma and Dongyu Qiu. A novel algorithm of optimistic unchoking for bittorrent. 2008.

[26] Green Matthew. Napster opens pandora's box: Examining how file-sharing services threaten the enforcement of copyright on the internet. page 63:799. Ohio State Law Journal, 2002.

[27] Robert Menta. Judge to decide on kazaa suit, 2002. http://www.mp3newswire.net/stories/2002/kazaatrial.html.

[28] Philip Miller. Tcp/ip explained, March 1, 1997. Digital Press.

[29] Bradley Mitchell. edonkey/overnet p2p file sharing client, 1999. http://compnetworking.about.com/od/p2ppeertopeer/p/overnetedonkey.htm.

[30] R. Morris. Tcp behavior with many flows. ICNP, 1997.

[31] T. S. E. Ng, Y.-H. Chu, S. G. Rao, K. Sripanidkulchai, and H. Zhang. Measurement- based optimization techniques for bandwidth-demanding peer-to-peer systems. IEEE INFOCOM, 2003.

[32] OpenNap. The napster protocol, April 7, 2000. http://opennap.sourceforge.net/napster.txt.

[33] A. Parker. The true picture of peer-to-peer filesharing, February 2007. http://www.cachelogic.com.

[34] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in bittorrent. USENIX Symposium on Networked Systems Design And Implementation, April 2007.

[35] F. L. Piccolo, G. Neglia, and G. Bianchi. The effect of heterogeneous link capacities in bittorrent-like file sharing systems. HOT-P2P, 2004.

[36] TechMetrix Research. P2p and its impact on the enterprise, Dec, 2001. http://www.ebizq.net/topics/real time enterprise/features/2303.html page=4.

[37] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. 2001.

[38] S. Saroiu, K. Gummadi, R. Dunn, and S. Gribble. A measurement study of peer-to-peer file sharing systems. Multimedia Computing and Networking MMCN, 2002.

[39] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. Gribble, and H. M. Levy. An analysis of internet content delivery system. the Fifth Symposium on Operating System Design and Implementation OSDI, 2002.

[40] R. Thommes and M. Coates. Bittorrent fairness: Analysis and improvements. Workshop Internet, Telecom. and Signal Proc, December 2005.

[41] Y. Tian, D. Wu, and K. W. Ng. Modeling, analysis and improvement for bittorrent-like file sharing networks. IEEE INFOCOM, 2006.

[42] Liao W.-C., Papadopoulos F., and Psounis K. A peer-to-peer cooperation enhancement scheme and its performance analysis. pages Vol 1, No.7. JOURNAL OF COMMUNICATIONS, NOVEMBER/DECEMBER 2006.

[43] Wikipedia. Bitcomet, 2007. http://en.wikipedia.org/wiki/BitComet.

[44] Wikipedia. Emule, 2007. http://en.wikipedia.org/wiki/EMule.

[45] Wikipedia. File sharing, 2007. http://en.wikipedia.org/wiki/File sharing.

[46] Wikipedia. Peer-to-peer, 2007. http://en.wikipedia.org/wiki/Peer-to-peer.

[47] Wikipedia. Client-server, Dec, 2001. http://en.wikipedia.org/wiki/Client-server.

[48] Wikipedia. Bittorrent, February 2007. http://www.bittorrent.com/protocol.html.

[49] Donna Wolff. Networking definitions - peer-to-peer, Feb, 2004. http://searchnetworking.techtarget.com/sDefinition/0,,sid7 gci212769,00.html.

[50] X. Yang and G. de Veciana. Service capacity of peer to peer networks. IEEE INFOCOM, 2004.

[51] H. Zhang, D. Towsley, and W. Gong. Tcp connection game: A study on the selfish behavior of tcp users. IEEE ICNP, 2005.