View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by IntechOpen

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

21

Performance Improvements of Peer-to-Peer File Sharing

Dongyu Qiu
Concordia University
Canada

1. Introduction

In recent years, *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*), P2P systems normally have much better scalability. The performance of 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 the P2P applications, file sharing has been one of the most popular. Traffic from P2P file sharing applications, such as Kazaa, Gnuttella, eDonkey, and BitTorrent (Cohen, 2003), has been dominating the Internet bandwidth in recent years. In this chapter, we will focus on the performance improvements of BitTorrent networks.

The performance of a BitTorrent network is affected by many factors. For example, how many pieces the served file is divided? How many neighbors a given peer has? Are peers cooperative or not? etc. In this chapter, we will try to improve the performance of a BitTorrent network from two aspects. Firstly, we assume that all peers in the network are cooperative, i.e., peers are willing to contribute by uploading. Under this assumption, we propose a stochastic model to study how to design an efficient P2P system. Secondly, we relax the cooperation assumption and study how to prevent selfish peers from free-riding. In BitTorrent, there are two built-in mechanisms to prevent free-riding. However, they are not very efficient. In this chapter, we will focus on one of the mechanisms called "optimistic unchoking" and discuss how its performance can be improved.

In a P2P network, if a peer is cooperative, it contributes to the network through uploading and hence it is important to efficiently utilize the upload bandwidth of each peer. The number of pieces that a given peer has is an important factor that affects the upload bandwidth utilization. For example, when a peer first enters the network, it has no pieces at all and hence can not upload to anyone. The upload bandwidth utilization is 0 in this case. On the other hand, when a peer has most of the pieces, it is very likely that it can upload to others and hence the utilization is close to 1. Motivated by this fact, we will propose a stochastic model to study the peer distribution with regards to the number of pieces that a peer has. More specifically, we are interested in P_i, which is the probability that a random peer has i pieces, where $0 \leq i \leq N$ and N is the total number of pieces of the served file. Note that in BitTorrent, peers that have the whole file are called seeds, while other peers are called downloaders. By numerically solving the proposed model, we will be able to gain interesting insight on how the performance of a P2P file sharing network is affected by different parameters such as the piece numbers of the

file, the number of neighbors of a peer, and the seed departure rate etc. We will then provide some useful guidelines on how to design an efficient P2P system based on our results.
In a real network, however, peers may not always be cooperative. The so-called 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 and another mechanism called "optimistic unchoking" to find upload bandwidth information about neighbors. However, both "Tit-for-Tat" and "optimistic unchoking" have some drawbacks. For example, a peer can adjust the upload rate and the number of uploads to take advantage of Tit-for-Tat. It has also been shown that a free-rider can obtain at least one fifth download rate of a normal peer just from optimistic unchoking. Hence, the two mechanisms can not efficiently prevent free-riding. In this chapter, we will propose a new optimistic unchoking algorithm for BitTorrent. Theoretical analysis and simulation results will be provided to show the effectiveness of our new mechanism.
The chapter is organized as follows. In Section 2, we will study how the network efficiency is affected by different parameters and discuss some guidelines on how to design an efficient P2P file sharing network. In Section 3, we will propose a new optimistic unchoking algorithm for BitTorrent and show how the new mechanism can improve the performance of the network.

2. The Efficiency of Peer-to-Peer File Sharing

In P2P file sharing, an interested file is divided into many pieces. The size of each piece ranges from several hundred kilobytes to several megabytes. When a new peer joins the network, it begins to download pieces from other peers. As long as it obtains one piece of the file, the new peer can start to serve other peers by uploading pieces. Since peers are downloading and uploading at the same time, when the network becomes large, although the demands increase, the service provided by the network also increases. Hence, the performance of the P2P network scales very well.
BitTorrent has been one of the most popular P2P file sharing applications and has attracted a lot of research attentions. While early work on P2P systems has mainly focused on system design and traffic measurements (Ng et al., 2003; Ripeanu, 2001; Ripeanu et al., 2002), some recent research has emphasized on performance modeling. In (Ge et al., 2003), a closed queueing system is used to model a general P2P file sharing system and basic insights on the stationary performance are provided. In (Clevenot & Nain, 2004; Clevenot et al., 2005), a stochastic fluid model is used to study the performance of P2P web cache (SQUIRREL) and cache clusters. In (de Veciana & Yang, 2003; Yang & de Veciana, 2004), 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 (Susitaival et al., 2006), a spatio-temporal model is proposed to analyze the resource usage of P2P systems. In (Susitaival & Aalto, 2006), an approximation for the life time of a chunk in BitTorrent is proposed. (Guo et al., 2005) presents an extensive trace analysis and modeling study of BitTorrent-like systems. In (Tian et al., 2006), the authors studied the behavior of peers in BitTorrent and also investigated the file availability and the dying-out process. In (Qiu & Srikant, 2004), a simple fluid models is proposed to study the performance and scalability of BitTorrent-like P2P systems.
In P2P file sharing networks, the upload bandwidth of each peer is a very important resource of the network. The efficient use of it will impact the system performance significantly. However, little research has been done in this area. In this section, we will focus on the efficiency

of P2P file sharing. More specifically, we will use a stochastic model to study how the efficiency is related to different network parameters, such as, the number of pieces, the number of neighbors, and the seed departure rate etc.
This section is organized as follows. In Section 2.1, we will propose a stochastic model for BitTorrent-like P2P file sharing networks and use this model to study the network efficiency. Simulation and numerical results of our model will be shown in Section 2.2 and we will also discuss the impacts of the results. Finally, we conclude in Section 2.3.

2.1 Stochastic Model

In a P2P network, each peer contributes to the network through uploading and hence it is important to efficiently utilize the upload bandwidth of each peer. The number of pieces that a given peer has is an important factor that affects the upload bandwidth utilization. For example, when a peer first enters the network, it has no pieces at all and hence can not upload to anyone. The upload bandwidth utilization is 0 in this case. On the other hand, when a peer has most of the pieces, it is very likely that it can upload to others and hence the utilization is close to 1. Motivated by this fact, we will next propose a stochastic model to study the peer distribution with regards to the number of pieces that a peer has. More specifically, we are interested in P_i, which is the probability that a random peer has i pieces, where $0 \leq i \leq N$ and N is the total number of pieces of the served file. Note that in BitTorrent, peers that have the whole file are called seeds, while other peers are called downloaders.
From real trace measurements, it has been observed that a BitTorrent-like P2P file sharing network has three phases (Yang & de Veciana, 2004), a growing phase, a stabilizing phase, and a decaying phase. The stabilizing phase is normally the one that most of the downloads take place and hence it is the one that determines the performance of the system. In this section, we consider a large P2P network that is in the steady state. Hence, the peer distribution $\{P_i\}$ is not changing with time. When a new peer first enters the network, it randomly picks up L peers as its neighbors. For the simplicity of analysis, we assume that the number of neighbors of each peer is fixed at L. For a given peer with i pieces, we assume that these pieces are chosen randomly from the set of all pieces of the file. This is a reasonable assumption because BitTorrent-like systems take a rarest first piece selection policy when downloading. Hence, it is unlikely that one piece has significantly more copies than other pieces. We also assume that these pieces are chosen independently with other peers. In P2P networks, a peer is normally downloading from many neighbor peers at the same time. Hence, a single neighbor's effect on the given peer's pieces can be neglected and it is reasonable to assume the independence between peers. Since we are interested in the efficient use of upload bandwidth, for simplicity of analysis, we assume all peers have the same upload bandwidth and the download bandwidth is unlimited. We use a discrete time model and without loss of generality, we assume that the upload bandwidth of a peer is one piece per time slot.
Note that although our model is relatively simple, it has all the important features of a typical P2P network and we expect the model can shed some light on further research of P2P file sharing efficiency. Next, we will start the analysis by focusing on only two peers in a neighborhood first. Assume that peer A has i pieces and peer B is a neighbor of peer A with j pieces. We define $F(i,j)$ to be the probability that peer B has no pieces that peer A is interested, i.e., peer A has all pieces of peer B, then

$$F(i,j) = \begin{cases} 0, & i < j \\ \frac{\binom{N-j}{i-j}}{\binom{N}{i}}, & i \geq j \end{cases}$$

In BitTorrent-like systems, when a peer obtains a new piece, it will update this information with its neighbors and hence a peer knows what pieces its neighbors have. At the beginning of a time slot, a peer will send requests for pieces to its neighbors if a neighbor has pieces that the peer is interested in. At the same time, the peer will also receive requests from its neighbors. If the peer receives more than one requests, it will randomly pick up one request to fulfill. Note that in a real BitTorrent network, peers will fulfill requests according to a built-in incentive mechanism. How the incentive mechanism will affect the performance is out of the scope of this chapter. Again, let's consider two peers A and B, where A has i pieces and B has j pieces. Under the condition that B has pieces that A is interested in, A will send a request to B. But the request from A may not be fulfilled since B also receives requests from other peers and B will randomly pick up one to fulfill. Let X be the number of requests that peer B receives besides A's request. Then X is a Binomial random variable with parameters $L-1$ and q_j, where $L-1$ is the maximum number of requests B could receive besides A's request and q_j is the probability that a randomly picked neighbor of B sends request to B. We have

$$q_j = \sum_{k=0}^{N} P_k(1 - F(k, j)).$$

The probability distribution of the random variable X is then

$$\mathbb{P}\{X = k\} = \binom{L-1}{k} q_j^k (1-q_j)^{L-1-k},$$

for $k = 0, \cdots, L-1$. So, when peer A sends a request to B, the probability that the request is fulfilled is

$$\begin{aligned}
G_j &= \sum_{k=0}^{L-1} \frac{1}{k+1} \mathbb{P}\{X = k\} \\
&= \sum_{k=0}^{L-1} \frac{1}{k+1} \binom{L-1}{k} q_j^k (1-q_j)^{L-1-k} \\
&= \sum_{k=0}^{L-1} \frac{(L-1)!}{(k+1)!(L-1-k)!} q_j^k (1-q_j)^{L-1-k} \\
&= \frac{1}{Lq_j} \sum_{k=0}^{L-1} \frac{L!}{(k+1)!(L-1-k)!} q_j^{k+1} (1-q_j)^{L-1-k} \\
&= \frac{1}{Lq_j} \sum_{k=0}^{L-1} \binom{L}{k+1} q_j^{k+1} (1-q_j)^{L-1-k} \\
&= \frac{1}{Lq_j} \sum_{m=1}^{L} \binom{L}{m} q_j^m (1-q_j)^{L-m} \\
&= \frac{1-(1-q_j)^L}{Lq_j}
\end{aligned}$$

The probability that peer A can download a piece from a random neighbor is then

$$S_i = \sum_{j=0}^{N} P_j(1 - F(i, j)) G_j.$$

In BitTorrent-like systems, at any given time slot, a peer only sends one request for a given piece. Hence, if a peer has i pieces, the maximum number of requests it can send is then $D = \min(L, N - i)$. The number of pieces that it downloads in the same time slot is then a Binomial random variable with parameters D and S_i. Define $r_{i,k}$ to be the probability that a peer with i pieces downloads k pieces in the current time slot, then

$$r_{i,k} = \binom{D}{k} S_i^k (1 - S_i)^{D-k}, \tag{1}$$

where $i = 0, \ldots, N-1$ and $k = 0, \ldots, D$. The average number of pieces that a peer with i pieces can download in a given time slot is then

$$d_i = \sum_{k=1}^{\min(L,N-i)} k r_{i,k}.$$

We also call d_i the average download rate of the peer. When the system is in the steady state, the peer distribution should not change with time. So, we have,

$$P_i = \sum_{k=0}^{\min(i,L)} P_{i-k} r_{i-k,k} \tag{2}$$

for $i = 1, \ldots, N$. The case $i = 0$ is a special one that is related to the peer arrival rate and we deal with it separately as following. When a peer has downloaded all the pieces, it becomes a seed. After a peer becomes a seed, it will stay in the system for an exponentially distributed time. Since we use a discrete time model, we define γ to be the probability that a seed will leave the system in the current time slot. Then $P_N \gamma$ is the total seed departure rate. When the system is in steady state, it should equal to the peer arrival rate. So, we have

$$P_0 = P_0 r_{0,0} + P_N \gamma. \tag{3}$$

Solving Eqs. (1)(2)(3), we can then obtain the peer distribution $\{P_i\}$. Note that it is hard to get a closed form peer distribution. However, the equations can easily be numerically solved and the results will be discussed in Section 2.2.

Once we know the peer distribution, we can use it to study important performance such as the average download time of a P2P network. Let the peer arrival rate be λ and the average number of peers in the system be M. Define T to be the average time a peer stays in the system and T_d to be the average download time of a peer (i.e., the time from a peer enters the system to it becomes a seed). Applying the Little's Law to the whole system, the seeds, and the downloaders respectively, we have

$$M = \lambda T$$
$$M P_N = \lambda \frac{1}{\gamma}$$
$$M(1 - P_N) = \lambda T_d$$

It is easy to see that

$$T = \frac{1}{P_N \gamma} \tag{4}$$

$$T_d = \frac{1 - P_N}{P_N \gamma} \tag{5}$$

Next, we will study how the system performance can be affected by different parameters such as the number of pieces N, the number of neighbors L, and the seed departure rate γ etc.

2.2 Simulation and Numerical Results

To validate the proposed stochastic model, we first simulate a BitTorrent-Like system and compare the simulation results with the peer distribution derived from the stochastic model. Our simulation follows the same assumptions as the stochastic model. Time is slotted and the upload bandwidth of a peer is one piece per time slot. In the simulation, the served file is divided into $N = 200$ pieces. Each peer is connected to $L = 20$ neighbor peers. Peers arrive to the network according to a Poisson process with the average arrival rate λ and the seed departure rate is $\gamma = 0.01$. This is a quite typical BitTorrent system setting since when a peer first enters the network, the default number of peers it gets from the tracker (Cohen, 2003) is normally 20. In Fig. 1 and 2, we show the simulation results of piece distribution for $\lambda = 20$ and $\lambda = 50$ respectively. In both cases, we see that the simulation results match well with the theoretical results derived from the stochastic model. Furthermore, when the peer arrival rate is larger ($\lambda = 50$), the simulation result is closer to the theoretical one. This is because our model assumes a very large P2P network. When λ is large, there will be more peers in the network and hence the stochastic model is more accurate.

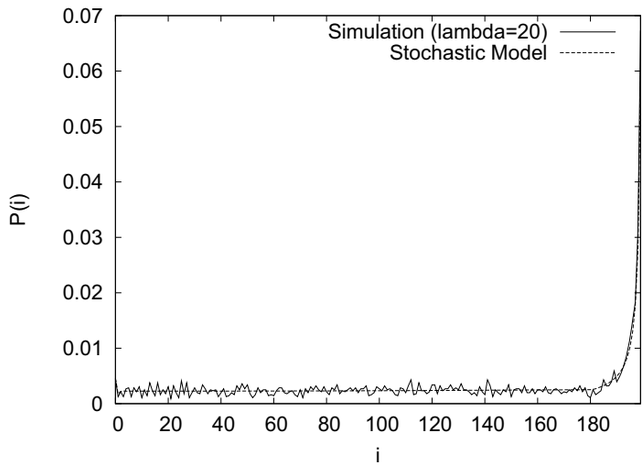

Fig. 1. Piece Distribution ($N = 200, \lambda = 20$)

In Figs. 1 and 2, the peer distribution is with regards to the number of pieces i that a peer has. We see when $i \leq 180$, peers are almost uniformly distributed. And when $i > 180$, the probability that a peer has i pieces increases when i increases. That is because when a piece has most pieces ($i > 180$), it can't fully utilize all of its neighbors anymore and hence its download rate decreases. We call this the end-game effect. In BitTorrent, there is even an end-game mode to deal with this. The details about the end-game mode is out of scope of this chapter. In Fig 3, we also show the average download rate of a peer with i pieces, we clearly see that when $i \leq 180$, the download rate is almost a constant. This explains the reason why in BitTorrent, the download rate of a peer can normally maintain at a high value for most part of the download process. When $i > 180$, however, the download rate keep decreasing and that also explains the result of Figs 1 and 2. Note that starting from Fig 3, we will only show the numerical results derived from the stochastic model to illustrate the figures more clearly.
In Fig. 4 and Fig. 5, we change the piece number from 200 to 100 and keep other parameters the same. We observe similar results as in the case of $N = 200$. However, since the piece number is 100 now, the end-game effect starts from $i = 80$.

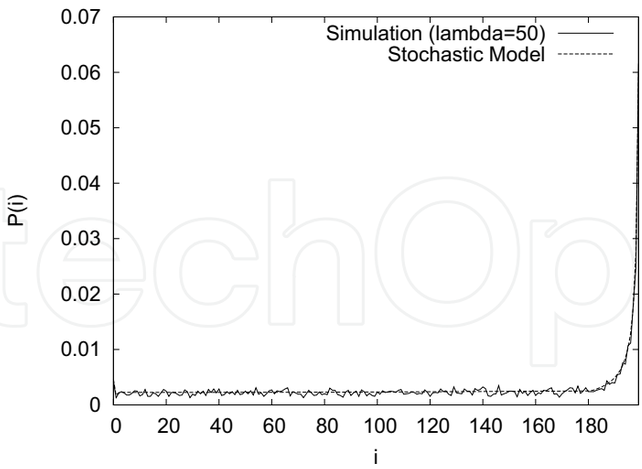

Fig. 2. Piece Distribution ($N = 200, \lambda = 50$)

In Fig 6, we keep $N = 200, L = 20$ and increase γ from 0.01 to 0.9. It shows the normalized download time $\frac{Td}{N}$ as a function of the seed departure rate γ. When γ increases, the download time also increases because there are fewer seeds in the system. However, if $\gamma > 0.3$, when γ increases, the download time doesn't change much. This tells us that the seed departure rate affects the system performance but it is not significant once γ is greater than some threshold. Note that when γ is too large, it may happen that no single seed in the system and hence cause the survivability problem of the network.
In Fig 7, we keep $L = 20, \gamma N = 2$ and increase the piece number N from 25 to 300. We see that when N increases, the average download time decreases as expected since larger N means a peer is more likely to upload to its neighbors. Note that we are keeping γN a constant instead of just γ because when N increases, the length of each time slot decreases (assuming the file size is a constant), hence we need to adjust γ accordingly.
In Fig 8, we keep $N = 200, \gamma = 0.01$ and increase the neighbors from $L = 2$ to $L = 20$. We see that when $L = 6$, the download time is the smallest. The explanation is that when L is too small, the probability that a peer can upload to its neighbor is small and hence not very efficient. However, when L is too large, the end game effect will be significant and will increase the download time. In Fig 9, we show the effect of the neighbor number when the parameters are changed to $N = 100, \gamma = 0.02$. Again, we see that the download time increases when L is too small or too large. In this case, $L = 4$ gives the smallest download time.

2.3 Conclusion

In this section, we propose a stochastic model to analyze the efficiency of P2P file sharing. We also verify the model through simulations. By solving the model numerically, we are able to gain some important insights on how the performance of P2P file sharing is affected by different parameters and based on these results, we are able to obtain some guidelines on how to design an efficient P2P file sharing system. Our contributions are: 1. The end game stage affects the performance significantly. To improve the performance, it is important to alleviate the end game effect. 2. If not considering the survivability, the seed departure rate doesn't affect the performance significantly when it is large enough. Hence, as long as we

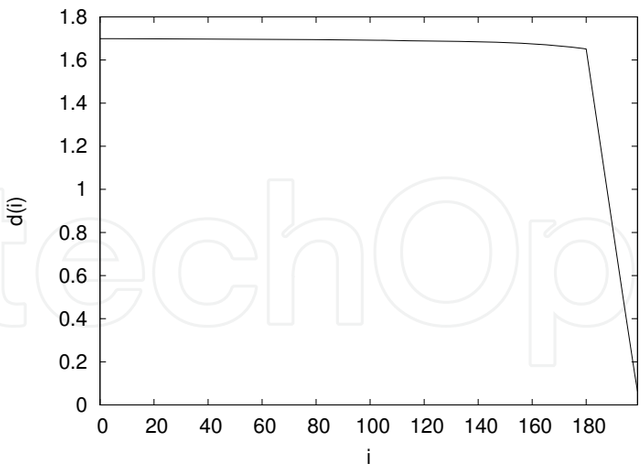

Fig. 3. Download Rate Distribution ($N = 200$)

have at least one seed in the system, it is not necessary to ask seeds to stay in the system for a long period of time. 3. Too many neighbors may affect the performance adversely since it cause more severe end game effect. Hence, it is important to choose a reasonable number of neighbors.

3. The Optimistic Unchoking Algorithm of BitTorrent

In a BitTorrent network, a shared file is divided into many pieces (the default size of a piece is 256KB). Peers that have the whole file are called seeds, while other peers with partial or none of the file are called downloaders. When a peer first joins the network, 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) (Cohen, 2003) 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

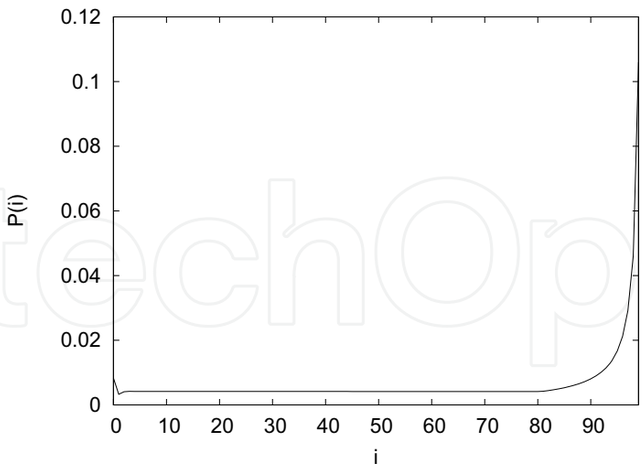

Fig. 4. Piece Distribution ($N = 100$)

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.

However, optimistic unchoking also introduces new problems. One of them is the free-riding. In (Qiu & Srikant, 2004), 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 section, 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.

This section is organized as following. In Section 3.1, we will discuss related work. In Section 3.2, a new optimistic unchoking algorithm for BitTorrent is proposed. In Section 4, a stochastic model is proposed to study the new algorithm. In Section 4.1, we present the simulation results. Finally, we draw the conclusion in Section 4.2.

3.1 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 (Yang & de Veciana, 2004), a branching process is used to study the service capacity of BitTorrent-like P2P file sharing in the transient regime and a simple Markovian

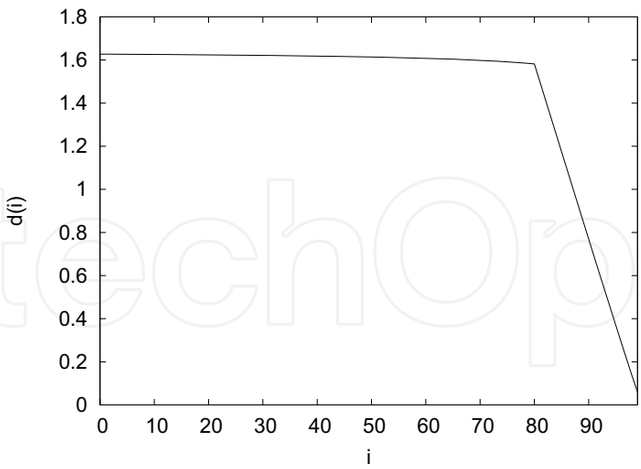

Fig. 5. Download Rate Distribution ($N = 100$)

model is presented to study the steady-state properties. In (Qiu & Srikant, 2004), a simple fluid models is proposed to study the performance and scalability of BitTorrent-like P2P systems. In (Tian et al., 2006), 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 (Qiu & Srikant, 2004), the effect of "Tit-for-Tat" is studied when selfish peers are able to adjust their uploading bandwidth. In (Zhang et al., 2007), 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 (Piatek et al., 2007), 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.

Our work in this chapter differs from the above-mentioned work in the following respect: we focus on the optimistic unchoking instead of the built-in incentive mechanism of BitTorrent. From our discussion above, we see that the optimistic unchoking plays an important role in BitTorrent. However, the current optimistic unchoking algorithm used in BitTorrent is not effective. Our theoretical and simulation results show that a well-designed optimistic unchoking algorithm can improve the performance significantly. In addition, unlike the incentive mechanism which requires all peers to implement the same mechanism for it to work well, our proposed optimistic unchoking algorithm can be deployed progressively. A peer can improve its performance by using our algorithm no matter other peers use the same algorithm or not. Our proposed algorithm can also co-exist with any incentive mechanisms, no matter it is the built-in "Tit-for-Tat" or other mechanisms proposed in the literature. In this sense, our proposed optimistic unchoking algorithm can be seen as a good complement to the incentive mechanism.

3.2 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

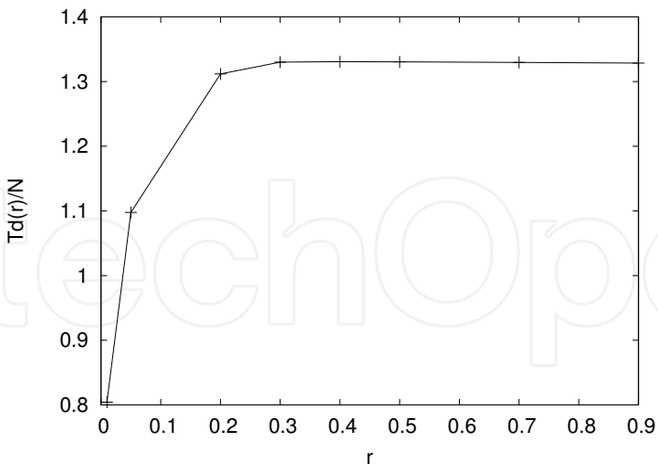

Fig. 6. The effect of seed departure rate

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. 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 (Qiu & Srikant, 2004), 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 to 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

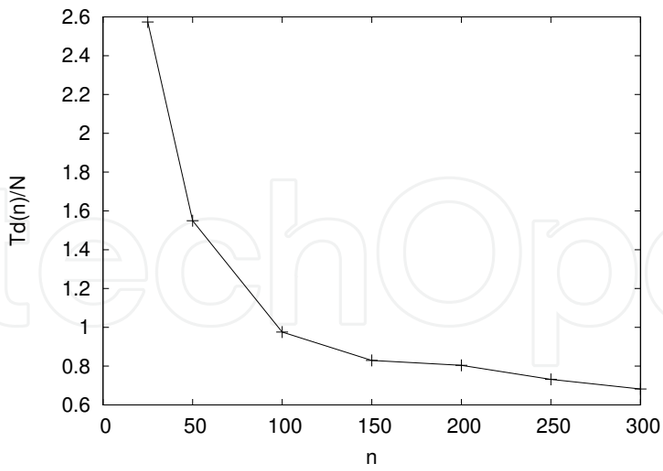

Fig. 7. The effect of piece number

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 us 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 happens, 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 choose 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

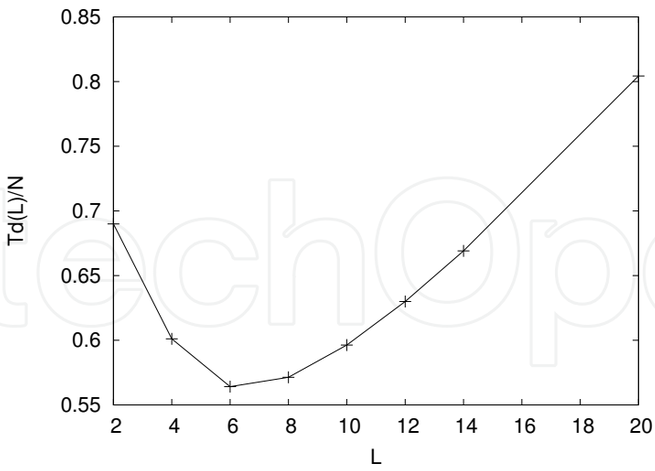

Fig. 8. The effect of neighbor number ($N = 200$)

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.

4. Stochastic Model

In Section 3.2, 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 3.2.

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.

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.

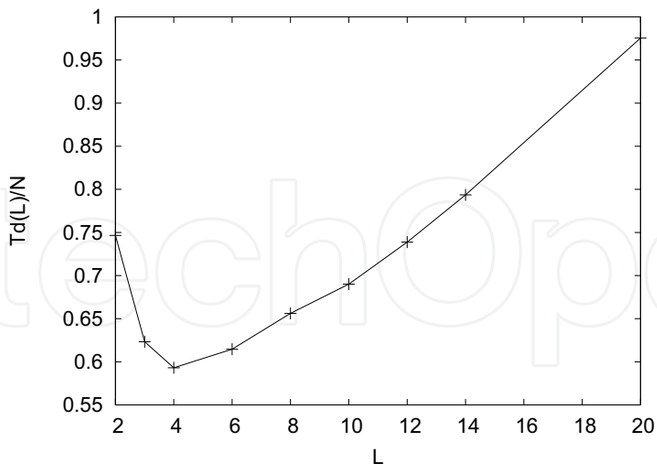

Fig. 9. The effect of neighbor number ($N = 100$)

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

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

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 $u(1-p)$.

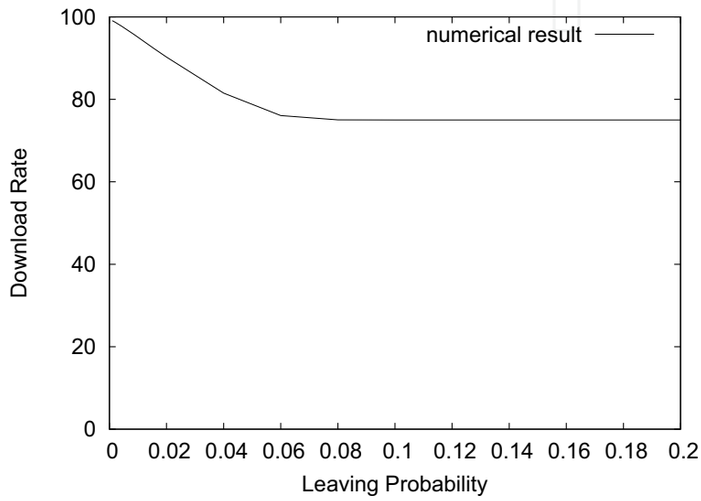

Fig. 10. The average download rate of a normal peer

In Fig. 10, 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.

4.1 Simulation Results

We have also done extensive simulations to evaluate the proposed algorithm. In the first simulation, we 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. Each peer in the network will randomly choose $L = 20$ as its neighbors when it joins the network. Peers arrive following a *Poisson process* with parameter $\lambda = 5.0$ and peers in the network leave randomly according to *Exponential departure* with parameter γ. In our simulation, when a peer finishes its download, it becomes a seed and provide uploading to others.

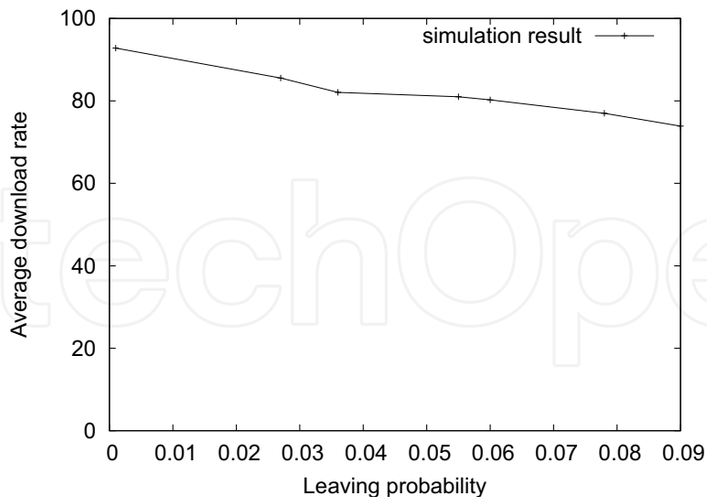

Fig. 11. Simulation: the average download rate of a normal peer

In Fig. 11, 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 a peer will use optimistic unchoking to test its new neighbor more often. Due to the possibility that newly-arrived peers may be free-riders, a normal peer who is already in the network may lose its chance to get the download via the optimistic unchoking. That is the 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 numerical result. This is because in the stochastic model, we assume that a free-rider can be detected with just one unchoke. While in real BitTorrent networks, it may need several unchokes to identify a free-rider.

In the second simulation, 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. 12, 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. 13, 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.

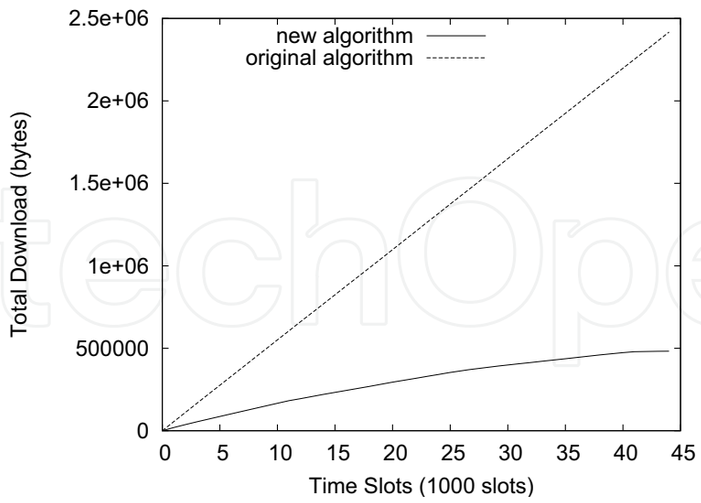

Fig. 12. The total download of a free-rider

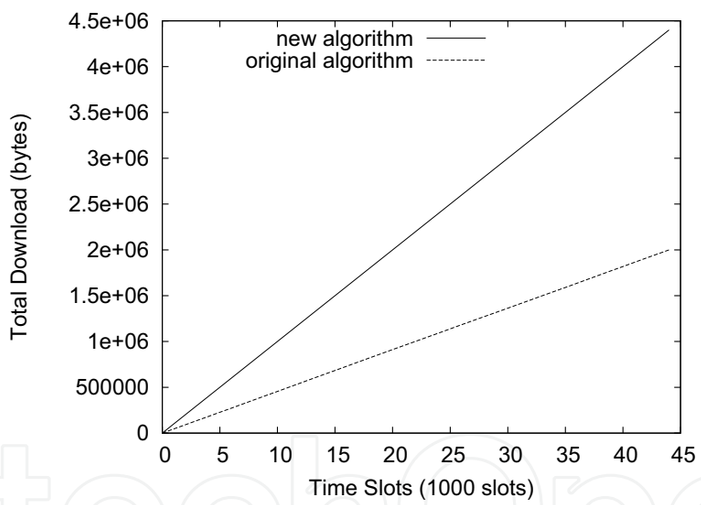

Fig. 13. The total download of a normal peer

4.2 Conclusion

In this section, 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.

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.

5. References

Clevenot, F. & Nain, P. (2004). A Simple Fluid Model for the Analysis of the Squirrel Peer-to-Peer Caching System, *Proceedings of IEEE INFOCOM*.

Clevenot, F., Nain, P. & Ross, K. (2005). Stochastic Fluid Models for Cache Clusters, *Performance Evaluation* **59**(1): 1–18.

Cohen, B. (2003). Incentives Build Robustness in BitTorrent, *Proceedings of the First Workshop on the Economics of Peer-to-Peer Systemas*, Berkeley, CA, USA.

de Veciana, G. & Yang, X. (2003). Fairness, incentives and performance in peer-to-peer networks, *the Forty-first Annual Allerton Conference on Communication, Control and Computing, Monticello, IL*.

Ge, Z., Figueiredo, D. R., Jaiswal, S., Kurose, J. & Towsley, D. (2003). Modeling peer-peer file sharing systems, *Proceedings of IEEE INFOCOM*.

Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X. & Zhang, X. (2005). Measurements, analysis, and modeling of bittorrent-like systems, *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement (IMC)*, pp. 35–48.

Ma, Z. & Qiu, D. (2009). A Novel Optimistic Unchoking Algorithm for BitTorrent, *Proceedings of Consumer Communications and Networking Conference*, Las Vegas, NV, USA.

Ng, T. S. E., Chu, Y.-H., Rao, S. G., Sripanidkulchai, K. & Zhang, H. (2003). Measurement-Based Optimization Techniques for Bandwidth-Demanding Peer-To-Peer Systems, *Proceedings of IEEE INFOCOM*.

Piatek, M., Isdal, T., Anderson, T., Krishnamurthy, A. & Venkataramani, A. (2007). Do incentives build robustness in bittorrent?, *Proc. of USENIX Symposium on Networked Systems Design & Implementation*.

Qiu, D. & Srikant, R. (2004). Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks, *Proceedings of ACM SIGCOMM*.

Ripeanu, M. (2001). Peer-to-peer architecture case study: Gnutella network, *Technical report*, University of Chicago.

Ripeanu, M., Foster, I. & Iamnitchi, A. (2002). Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design, *IEEE Internet Computing Journal* **6**(1).

Susitaival, R. & Aalto, S. (2006). Modelling the population dynamics and the file availability in a bittorrent-like p2p system with decreasing peer arrival rate, *International Workshop on Self-Organizing Systems (IWSOS)*, pp. 34–48.

Susitaival, R., Aalto, S. & Virtamo, J. T. (2006). Analyzing the dynamics and resource usage of p2p file sharing by a spatio-temporal model., *International Conference on Computational Science (4)*, pp. 420–427.

Tian, Y., Wu, D. & Ng, K. W. (2006). Modeling, Analysis and Improvement for BitTorrent-Like File Sharing Networks, *Proceedings of IEEE INFOCOM*.

Yang, X. & de Veciana, G. (2004). Service Capacity of Peer to Peer Networks, *Proceedings of IEEE INFOCOM*.

Zhang, H., Neglia, G., Towsley, D. & Presti, G. L. (2007). On unstructured file sharing networks, *Proceedings of IEEE INFOCOM*.

Multimedia

Edited by Kazuki Nishi

ISBN 978-953-7619-87-9

Hard cover, 452 pages

Publisher InTech

Published online 01, February, 2010

Published in print edition February, 2010

Multimedia technology will play a dominant role during the 21st century and beyond, continuously changing the world. It has been embedded in every electronic system: PC, TV, audio, mobile phone, internet application, medical electronics, traffic control, building management, financial trading, plant monitoring and other various man-machine interfaces. It improves the user satisfaction and the operational safety. It can be said that no electronic systems will be possible without multimedia technology. The aim of the book is to present the state-of-the-art research, development, and implementations of multimedia systems, technologies, and applications. All chapters represent contributions from the top researchers in this field and will serve as a valuable tool for professionals in this interdisciplinary field.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Dongyu Qiu (2010). Performance Improvements of Peer-to-Peer File Sharing, Multimedia, Kazuki Nishi (Ed.), ISBN: 978-953-7619-87-9, InTech, Available from: http://www.intechopen.com/books/multimedia/performance-improvements-of-peer-to-peer-file-sharing

INTECH

open science | open minds

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen