Stochastic Analysis of P2P File Sharing Systems

LIN, Minghong

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

September 2008

Thesis / Assessment Committee

Professor NG Wai Yin (Chair)

Professor LUI Chi Shing (Thesis Supervisor)

Professor NG Kam Wing (Committee Member)

Professor LAU Francis Chi Moon (External Examiner)

Abstract of thesis entitled:
Stochastic Analysis of P2P File Sharing Systems
Submitted by LIN, Minghong
for the degree of Master of Philosophy
at The Chinese University of Hong Kong in June 2008

File sharing is one of the most important applications in P2P networks. In this work, we propose stochastic models to analyze and improve the P2P file sharing systems. In the first part, we analyze the performance of the P2P file sharing systems under realistic settings. We first extend the results of the coupon system [Massoulie05] by providing a tighter performance bound. Then we generalize the coupon system by considering limited upload capacity and incentive mechanism. We illustrate the last-piece problem and show how to improve the performance. In the second part, we analyze and design an ISP-friendly protocol to reduce the inter-domain traffic generated by the traditional P2P file sharing systems. To carry out realistic study, we design and implement the protocol which is compatible with the current BitTorrent protocol and show how it can handle the "black-hole attack". Large scale experiments are carried out on the PlanetLab. This work suggests that stochastic modeling is a powerful tool in analyzing P2P systems.

中文摘要

文件共享是P2P網絡最重要的應用之一。本文中我們提出一個隨機模型來分析並提高P2P文件共享系統的性能。在前半部分，我們分析了P2P文件共享系統在現實條件下的性能。首先我們擴展了Coupon System[Massoulie05]的結果並得到更加緊致的界，然後我們把這個系統一般化以考慮上傳帶寬限制和激勵機制。我們闡述了”Last Piece Problem”以及如何提高系統性能。在後半部分，我們分析並設計了一個ISP-friendly的協議以減少傳統P2P文件共享系統引起的大量ISP之間的網絡流量。為了更加貼近現實，我們設計並實現了這個協議，這個協議與現有的BitTorrent協議兼容。我們展示了這個協議如何應付”black-hole attack”，並且在PlanetLab上進行了大規模的實驗。本文展示了隨機模型可以作為一個很強的工具對P2P系統進行分析。

Contents

List of Figures

Acknowledgement

I would like to start by expressing my thanks to my advisor, Prof. John C.S. Lui. He spends lots of time helping me in reaching my potential while maintaining my enthusiasm for the subject matter. From John I have been learning not only the serious attitude towards research, but also the profound insights and novel opinion to the challenging and exciting research field. I will also express my gratitude to him for introducing me to many talented researchers and giving me sincere advice for my future study. I would like to thank my co-advisor, Prof. Dah-Ming Chiu. He has been a true wealth of knowledge in my academic pursuits.The moments that I discussed with him were really wonderful. His advices on my research and comments on my work were priceless. I feel very fortunate I could study under John's and Dah-Ming's supervision during my M.Phil study.

I am grateful to Prof. Will Wai-Yin Ng, Prof. Kam-Wing Ng, as well as Prof. Francis Lau for taking time from their busy schedules to sit on my committee and feedback on my work. I would express my gratitude to the entire ANSR Lab too. The colleagues were not only friends but excellent critiques of my work. I really enjoyed having them as my groupmates, espeically those moments that we discuss together. I wish them the best of luck in their future study and career.

On the personal level, I would like to thank my parents and my brother. They have been incredibly understanding and supportive in my efforts to pursuit my dreams. Mom and dad, thank you for teaching me to never stop dreaming. Without your love, I would have been a very different person. I am grateful to you eternally.

Chapter 1

Introduction

In recent years, peer-to-peer (P2P) networks have emerged as a new paradigm for creating network applications. Recent network measurements have shown that P2P file-sharing applications constitute a large percentage of the network traffic. Also, P2P networks have a significant impact on the way new network services are designed. Unlike the traditional client-server computing paradigm, P2P networks allow individual user (or peer) to play the role of a client and server at the same time. Therefore, peers in a P2P network can help other peers in file searching, file lookup, as well as file transfer.

File sharing is one of the most important applications in P2P networks. In general, a P2P file sharing application has a good scalability property due to its collaborative mechanism, which can be intuitively explained as follows: a file is first partitioned into many disjoint *pieces*. Each peer can get these pieces either from a server, or from other peers holding those pieces that it does not already have. Each peer offers upload service to other peers, and in return, each peer tries to obtain a missing piece so as to maximize its ability to serve others hence also the service it will receive. By coupling the service each peer can receive to its contribution to others, P2P file sharing applications successfully make each peer to play a role of a server and a client at the same time. Therefore, as the number of peers increases, the service capacity of the whole system increases as well. File sharing application is implemented in P2P networks such as eDonkey, KaZaA, and it is the core functionality of the popular BitTorrent (BT) [2] protocol.

The work by the authors in [34] suggest that P2P file sharing systems (e.g., BT networks) is efficient in the sense that as the demand for the file increases, the service capacity increases as well. However, it is not completely understood which aspects of the system are critical to maintain the scalability property. The authors in [30] use a fluid model to represent the BT file swarming protocol and derive a coarse approximation of the average file downloading time. Recently, a coupon model [27] is proposed to represent a generic file swarming system. The authors

analyze the system under the large population regime and show the file swarming system stabilizes around a finite equilibrium point and is indeed efficient. The results provide further support to the claim of [34], and that the system performs well under the flash crowd scenario, even when the rarest first piece selection policy is replaced by some random coupon selection policies. However, strong assumptions are made in [27], in particular, the authors assume that peers have infinite upload capacity (or relatively large as compare with the download capacity).

The first part of the thesis aims to provide a deeper understanding to the P2P file sharing protocols and the efficiency of BitTorrent-like file sharing system. We propose a simple *density dependent jump Markov process* to model the dynamics of a file sharing system, and we investigate the performance of the system under constraints on upload capacity, download capacity, peer selection policies (including random piece selection and coordinated matching). The contributions of this part of work are

- We generalize some of the results in the coupon system [27] and provide a tighter bound for performance measures such as the average file downloading time.
- We consider the *last-piece problem* and analytically show the improvement in performance when a file swarming system uses the forward error correction (FEC) [32] coding technique for file sharing.
- We relax the unlimited upload capacity assumption in [27], analyze the file swarming system under a more realistic setting and provide asymptotic bounds on the average file downloading time.
- We propose a stochastic model for an incentive-based file swarming system with coordinated matching, wherein piece exchange is only allowed when both peers are deemed to be useful to each other.

Extensive simulations are carried out to validate our models and to illustrate some interesting design guidelines.

Although it performs quite well, the file sharing application like BitTorrent introduces some challenging issues. Studies show that P2P applications account for over 60% of the traffic seen by an ISP [6]. Worse yet, pre-dominant of the traffic goes through the cross-ISP links since these applications do not distinguish between ISPs' boundaries. This not only presents significant traffic-engineering challenges to ISPs, but the large volume of cross-ISP traffic also implies an increasing congestion level and more important, high operating cost for ISPs.

ISPs have several options to deal with the above problem. One approach is to control the file distribution traffic via packet throttling. However, this is not an effective solution since applications can always use

dynamic port to bypass detection. Also, throttling discourages users within an ISP and these users may opt to switch to another ISP for service. Another approach for an ISP is to perform caching so as to limit the cross-ISP traffic. However, caching can be complicated since ISP needs to accurately determine which file to cache or replace. Not only caching requires additional infrastructure and cost, but also introduces legal problem to the ISPs due to the copyright issue.

Researchers propose some techniques to reduce the cross-ISP traffic. One is to select a single peer, called the "gateway peer", to connect to the external world [19]. This technique requires constant maintenance of the gateway architecture and one has to deal with the potential selfish behaviors of peers, e.g., the gateway peer may not want to upload to other internal peers within this ISP and thereby brings down the system. Another technique is to modify the tracker to return more *internal* peers when a peer requests a neighbor list [5]. This technique weakens the connectivity of the P2P network in order to reduce the cross-ISP traffic. A potential problem is that the topology could not evolve accordingly and this may degrade the downloading performance.

So in the second part of the thesis, we introduce an "*ISP-friendly file distribution protocol*" based on the BitTorrent protocol. The goal of the protocol is to reduce the cross-ISP traffic, maintain good file downloading performance and at the same time, do away with expensive infrastructure support. From each peer's point of view, all other peers could be classified into two categories: *internal peers* and *external peers*. Internal (external) peers are those peers which belong to the same (different) ISP as itself. The protocol relies on the following idea: downloading pieces from internal peers as many as possible, i.e., peers tend not to consider external peers if the piece is held by some internal peers. To illustrate it more formally, we call this the "*exploiting-the-locality principle*" (ELP). The ELP is: *never download information from external peers if there exist at least one copy of the information among the internal peers.* It is possible to modify the BT clients' interested behaviors to follow this principle without changing the topology of the BitTorrent network, so that make the peer adapt to new situation much more quickly than the topology maintenance approach. The contributions of this part of work are:

- We *analytically quantify* the merits when file distribution protocols follow the ELP. In particular, we derive the lower and upper bounds of incoming cross-ISP traffic under regular peer arrival (i.e., Poisson process) and bursty peer arrival (i.e., flash crowd).
- We propose and implement an ISP-friendly protocol on existing BitTorrent client software. It is compatible with the current BitTorrent protocol. We show that a client only needs to control the *incoming* cross-ISP traffic and the *outgoing* cross-ISP traffic will be reduced

accordingly.

- We carry out experiments and measurements on PlanetLab to demonstrate significant cross-ISP traffic reduction and good file downloading performance.
- We illustrate the *black-hole security attack* and show how the modified ISP-friendly protocol can overcome this problem.

□ **End of chapter.**

Chapter 2

A Stochastic Framework

2.1 Model Description

Let us consider a peer-to-peer file sharing system that distributes a given file $\mathcal{F}$ to a number of peers. The file is divided into K equal size chunks, the i^{th} chunk is denoted as $\mathcal{C}_i$, and $\mathcal{F} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots \cup \mathcal{C}_K$, with $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ for $i \neq j$. To download the file $\mathcal{F}$, a peer needs to download all K chunks from other peers in this P2P file sharing system. Let $\mathcal{F}_A$ be the set of chunks that peer A possesses. Peer A maintains a *bitmap* to denote which chunks they possess. Whenever peer A finishes the downloading of a new chunk, it will update its bitmap. Peer A can upload chunk $\mathcal{C}_k$ to others only after it has completely downloaded $\mathcal{C}_k$. New peers arrive to this system according to a Poisson process with an average rate λ. Using the BitTorrent's terminology, a peer that has at least one missing chunk of $\mathcal{F}$ is called a *leecher*, while a peer that has all K unique chunks of $\mathcal{F}$ is called a *seeder*. Note that, unlike the BitTorrent system which has at least one seeder to start the file distribution and serve the leechers, we assume that every newly arrived peer will initially obtain one chunk from a server before entering this system[1]. This initial chunk is randomly chosen by the server with equal probability $1/K$ for chunks $\mathcal{C}_1 \ldots \mathcal{C}_K$. When a peer finishes downloading all K chunks, the peer will depart immediately.

Similar to [27], we assume that this P2P file sharing is slotted in the sense that uploading (or downloading) a single chunk takes one slot time. The file distribution process in each time slot can be described as follow. At the beginning of every time slot, a peer, say A, will select $m \geq 1$ other peers in the system and fetches their bitmaps. Note that, the parameter m and the way it chooses these m peers will greatly affect the system performance, and we will further investigate this in later sections. Since the bitmap information can be greatly compressed, the transfer time of a bitmap is negligible compared to the transfer time of a chunk. Let

[1]This assumption is similar to the one made in [27]

peer B be one of these m peers. Upon receiving its bitmap, peer A can determine whether peer B is useful (i.e. peer B possesses at least one missing chunk of peer A, or $\mathcal{F}_B \setminus \mathcal{F}_A \neq \emptyset$). If no peer among these m selected peers is useful to peer A, then peer A will take no action but remain idle in the current time slot; otherwise, peer A will randomly select one of the useful peers to request a useful chunk for download. Assume the selected peer is B, then peer A will request one chunk which is uniformly chosen from the set of chunks possessed by peer B and are missing in peer A (i.e. a chunk $\mathcal{C}_k \subset \mathcal{F}_B \setminus \mathcal{F}_A$). Note that this can be viewed as a *blind chunk selection policy*, in contrast to the *rarest first policy* in the BitTorrent protocol by which peer A will select the chunk among $\mathcal{F}_B \setminus \mathcal{F}_A$ with the fewest number of copies among its neighbors [8]. As a result, peer B may receive *multiple* downloading requests. Based on the upload capacity constraint and service rule, peer B will choose one or more requests to satisfy (we will elaborate this in later sections). The transfer time of this chunk will take one time slot. At the end of a time slot, the process repeats.

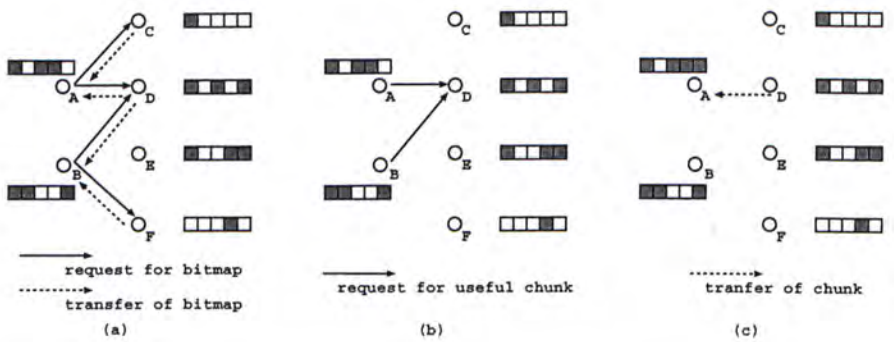


Figure 2.1: A simple illustration of a transfer dynamic within one time slot with $\mathcal{F} = \mathcal{C}_1 \cup \mathcal{C}_2 \cdots \cup \mathcal{C}_5$

Figure 2.1 illustrates the P2P file sharing model with $m = 2$. We have six peers: A, B, C, D, E and F. The file has five chunks and the shaded boxes represent the chunks that peers possess. For example, peer A has $\mathcal{C}_1, \mathcal{C}_3$ and $\mathcal{C}_4$. In Figure 2.1(a), peer A (peer B) requests bitmaps from peer C and D (peer D and F) and these peers reply with their respective bitmaps. Peer A determines that peer C is not useful while peer D is useful. Peer B, on the other hand, determines that both peer D and F are useful. Both peers select one peer for a chunk transfer and Figure 2.1(b) shows that both peer A and B choose D for the chunk transfer. Peer D receives two transfer requests, it randomly picks one peer to serve in this example, and it chooses peer A. Figure 2.1(c) shows that peer D transfers $\mathcal{C}_5$ which is requested by A. At the end of a time slot, peer A obtains $\mathcal{C}_5$ while peer B wastes one time slot.

The above model is in fact, quite general. For example, when one considers the case that $m = 1$ (or each peer just randomly chooses one peer to fetch the bitmap), and that there is no constraint on peers' upload capacity, then this becomes the model studied in the coupon replication system [27]. In this work, we generalize their model and study the performance of the system when $m \geq 1$, which means that each peer can first fetch multiple bitmaps from different peers but can choose at most one peer to request chunk transfer. Surprisingly, such a simple modification can improve the performance of the system to achieve a near optimal average file downloading time. Furthermore, we also relax the assumption of large or infinite upload capacity in [27]. This is in fact a very important step because for the current Internet, the bottleneck is usually not at the network core but rather at the network edge, and the upload capacity of an end host is indeed limited (e.g., ADSL system, cable system). Therefore, this capacity constraint model is in fact a more realistic representation for file sharing systems. In this uplink/downlink constrained system, we study two different uploading policies.

1. *Altruistic Uploading Service:* Under this policy, a peer will provide upload service to other peers regardless of whether these peers have provided upload service or not to other peers. In other word, this is a perfect collaborative system and it is similar to the "optimistic unchoking" feature in the BitTorrent protocol.

2. *Incentive Uploading Service:* Under this policy, a peer follows a given incentive mechanism similar to the "tit-for-tat" feature used in the BitTorrent protocol to decide on uploading.

Although our system model is a simple representation of some realistic P2P file sharing system (e.g., BitTorrent), it has already captured many essential features such as the *collaborative* upload and download, as well as incentive-based chunk exchange in P2P file sharing systems. In later sections, we will derive the performance of such system, and show why and how it can achieve good performance.

2.2 Altruistic File Sharing System with Download Constraint

In this section, we consider the file sharing system where each peer has a constraint in the download capacity and we place *no upper bound restriction* on the upload capacity. So at every time slot, each peer will first contact $m \geq 1$ other peers randomly in the system to get their bitmaps. If more than one peer are useful, it will randomly choose one to request a useful chunk. It is possible that a peer may get many downloading requests. Since we assume that there is no restriction on uploading

bandwidth, all requests will be satisfied. Also, due to the abundance of uploading bandwidth, there is no need to enforce incentive mechanism for data transfer. Lastly, it is important to note that when $m = 1$, this corresponds to the model described in coupon replication system [27].

2.2.1 Model Formulation

First we assume that all types of chunks in the system are uniformly distributed. This assumption can be guaranteed by the random chunk selection policy(as described in Section 2.1). We classify peers into different types according to the number of chunks it possesses. A peer holding i chunks is called a type i peer, for $i = 1, 2, \ldots, K-1$ ($i \neq K$ because a peer will immediately depart from the system when it finishes downloading all K chunks). After receiving a new chunk, a type i peer will become a type $(i+1)$ peer. Let $p_{i,j}$ denote the probability that a type j peer B is useful to a type i peer A. When $i < j$, it is clear that $p_{i,j} = 1$; When $i \geq j$, we have $p_{i,j} = 1 - \text{Prob}\{\mathcal{F}_B \subseteq \mathcal{F}_A\}$. Thus

$$p_{i,j} = \begin{cases} 1 & 1 \leq i < j \leq K-1, \\ 1 - \frac{C_i^j}{C_K^j} & 1 \leq j \leq i \leq K-1. (C_x^y \text{ is the binomial coefficient}) \end{cases} \tag{2.1}$$

Let $y_i(t)$ denote the number of type i peers in the system at time t. The total number of peers in the system at time t is $y(t) = \sum_{i=1}^{K-1} y_i(t)$. When a type i peer randomly picks another peer and requests its bitmap, the probability that this selected peer is useful is $q_i(t) = \sum_{j=1}^{K-1} p_{i,j} y_j(t)/y(t)$, $i = 1, 2, \ldots, K-1$.

Given the system state $\boldsymbol{Y}(t) = \{y_i(t)\}_{i \in \{1,\ldots,K-1\}}$, it is easy to verify that $(\boldsymbol{Y}(t))_{t \geq 0}$ is a Markov process taking its values in $\mathcal{Z}_+^{K-1}$ ($\mathcal{Z}_+^{K-1}$ is a $K-1$ dimensions vector with non-negative integer entities). Denoting by e_i the unit vector of $\mathcal{Z}_+^{K-1}$ whose i-coordinate equals 1, and with all other coordinates equal to zero, the non-zero transition rates of this Markov process are, for all $i \in \{1, ..., K-1\}$,

$$\begin{aligned} \boldsymbol{Y} &\longrightarrow \boldsymbol{Y} + e_1 && \text{with rate } \lambda, \\ \boldsymbol{Y} &\longrightarrow \boldsymbol{Y} - e_i + e_{i+1} && \text{with rate } y_i \left(1 - (1 - q_i)^m\right), i \in \{1, \ldots, K-2\} \\ \boldsymbol{Y} &\longrightarrow \boldsymbol{Y} - e_{K-1} && \text{with rate } y_{K-1} \left(1 - (1 - q_{K-1})^m\right). \end{aligned}$$

We analyze the system under a large population asymptotic regime. Note that this is a *density dependent jump Markov process* [22]. It converges to the solution of the differential equations

$$\frac{dy_i(t)}{dt} = \begin{cases} \lambda - y_1(t) \left[1 - (1 - q_1(t))^m\right] & i = 1, \\ y_{i-1}(t) \left[1 - (1 - q_{i-1}(t))^m\right] - y_i(t) \left[1 - (1 - q_i(t))^m\right] & i = 2, \ldots, K-1. \end{cases} \tag{2.2}$$

for some initial condition $\boldsymbol{Y}(0)$.

2.2.2 Steady State Analysis

In this section, we derive the average file downloading time for the above P2P file sharing system. We also extend our analysis to a file sharing system that provides forward error correction (FEC) service.

Altruistic File Sharing Without FEC

In this section we focus on the steady state performance and its *equilibrium point*. An equilibrium point is the point $\hat{\boldsymbol{Y}} = (y_1, y_2, \ldots, y_{K-1})$ such that if $\boldsymbol{Y}(t) = \hat{\boldsymbol{Y}}$, then $\boldsymbol{Y}(t') = \hat{\boldsymbol{Y}}$ for all $t' \geq t$. The necessary and sufficient condition for $\hat{\boldsymbol{Y}}$ to be an equilibrium point is $\frac{dy_i(t)}{dt} = 0$, for $1 \leq i \leq K-1$. Apply these conditions to Eq. (2.2), we have the following equations at the equilibrium point $\hat{\boldsymbol{Y}}$: $\lambda = y_i(1-(1-q_i)^m)$, $i = 1, 2, \ldots, K-1$.

Let T_i be the average sojourn time for type i peers, that is, the average time for a type i peer to receive a new chunk and become type $(i+1)$. One can derive this measure from the equilibrium point $\hat{\boldsymbol{Y}} = (y_1, \ldots, y_{K-1})$ by using Little's theorem [21]: $\lambda T_i = y_i$. Define $T = \sum_{j=1}^{K-1} T_j$ as the average file downloading time in the P2P file sharing system, we have $y_i/y = T_i/T$. Finally, one can obtain the following equations at the equilibrium point $\hat{\boldsymbol{Y}}$:

$$T_i = \frac{1}{1-(1-q_i)^m} \quad \text{and} \quad q_i = \sum_{j=1}^{K-1} \frac{T_j}{T} p_{i,j}, \quad \text{for } i = 1, 2, ..., K-1, \tag{2.3}$$

One can observe that T_i of Eq. (2.3) does not depend on λ. So even when the arrival rate λ is large and the number of peers in the system becomes very large, the average sojourn time T_i (and also T) will not be affected in the steady state. This is an important observation since this indicates that the P2P file sharing system has a good scaling property: when one increases the arrival rate, the performance will not degrade. Since T_i is the average sojourn time for type i peers, i.e. it takes on average, T_i unit of time slots to download the next chunk when a peer holds i chunks, let us explore the relationships among the T_i's at the steady state.

Lemma 1 *The sojourn time is an increasing sequence, i.e.* $1 \leq T_1 < T_2 < \cdots < T_{K-1}$.

Proof: According to Eq. (2.3) we have $q_i \leq 1$. Therefore, one can conclude that $T_i \geq 1$ for $i = 1, \ldots, K-1$. According to Eq. (2.1), when $i > i'$, $p_{i,j} \leq p_{i',j}$ holds for $j = 1, \ldots, K-1$ and $p_{i,j} < p_{i',j}$ holds for some j. So $q_i = \sum_{j=1}^{K-1} \frac{T_j}{T} p_{i,j} < \sum_{j=1}^{K-1} \frac{T_j}{T} p_{i',j} = q_{i'}$. Thus, we have $T_i > T_{i'}$ when $i > i'$. ∎

Lemma 2 *The upper and lower bounds of T_i are*

$$\frac{1}{1-\left[\left(\frac{1}{K-2+H_K}\right)\left(\frac{i}{K-i+1}\right)\right]^m}+O(K^{-2})<T_i<\frac{1}{1-\left[\left(\frac{1}{K-1}\right)\left(\frac{i}{K-i+1}\right)\right]^m},$$

where K is the number of chunks in $\mathcal{F}$ and H_K is the K^{th} harmonic number.

Proof: The sequence $\{t_j = T_j/T\}$ is increasing and the sequence $\{a_j = p_{i,j}\}$ in non-decreasing. From Chebyshev's sum inequality, we have

$$q_i > \frac{1}{K-1}\left(\sum_{j=1}^{K-1}\frac{T_j}{T}\right)\left(\sum_{j=1}^{K-1}p_{i,j}\right) = \frac{1}{K-1}\left(K-1-\sum_{j=1}^{i}\frac{C_i^j}{C_K^j}\right)$$
$$= 1-\left(\frac{1}{K-1}\right)\left(\frac{i}{K-i+1}\right) \text{ ("Concrete Mathematics" [16], p174.)}$$

One can apply it to Eq. (2.3) and obtain the upper bound of T_i as claimed. For the lower bound of T_i, let us first derive an upper bound of T, which is

$$T = \sum_{i=1}^{K-1}\frac{1}{1-(1-q_i)^m} \le \sum_{i=1}^{K-1}\frac{1}{q_i} < \sum_{i=1}^{K-1}\frac{(K-1)(K-i+1)}{K(K-i)-1}$$
$$= \frac{(K-1)^2}{K}+\frac{K^2-1}{K}\sum_{i=1}^{K-1}\frac{1}{Ki-1} = K-2+H_K+O(K^{-1}).$$

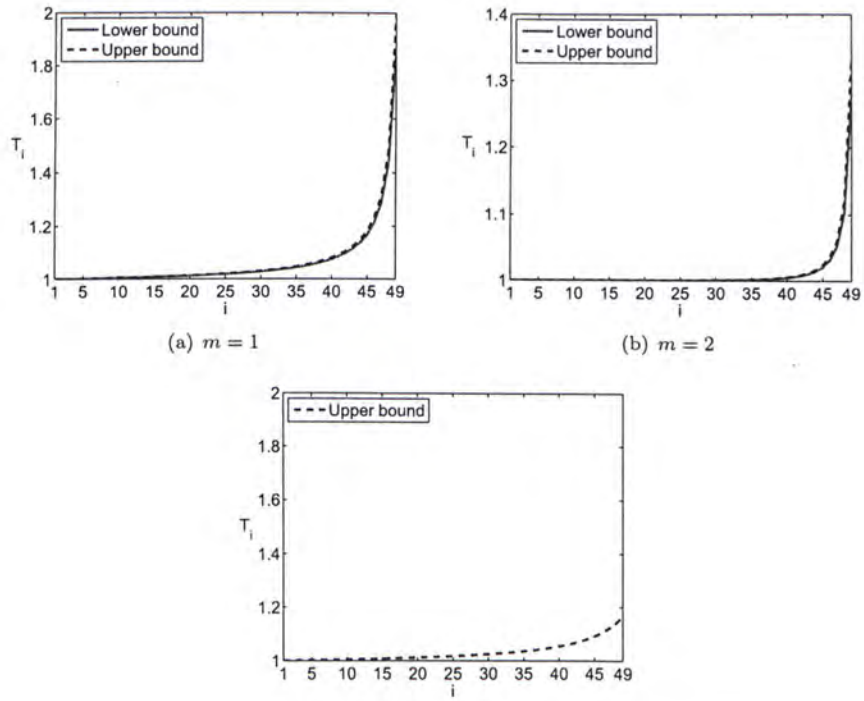
We can apply it to Eq. (2.3) to obtain an upper bound of q_i as

$$q_i = 1-\sum_{j=1}^{i}\left(\frac{T_j}{T}\right)\left(\frac{C_i^j}{C_K^j}\right) < 1-\sum_{j=1}^{i}\frac{C_i^j}{C_K^j}\left(\frac{1}{K-2+H_K+O(K^{-1})}\right)$$
$$= 1-\left(\frac{1}{K-2+H_K}\right)\left(\frac{i}{K-i+1}\right)+O(K^{-2}).$$

With this upper bound of q_i, one can substitute it to Eq. (2.3) to obtain the lower bound of T_i as claimed. ∎

Remark: The importance of the above two lemmas is that one can use them to understand the "*last-piece*" problem in P2P file sharing systems. i.e. how long it takes for a peer to receive the last few chunks of the file since it gets increasingly more difficult to find other peers that can help.

To illustrate this issue, let us consider the upper and lower bounds of T_i for a file with $K = 50$ chunks. The scenario is illustrated in Fig.2.2(a) and Fig. 2.2(b). There are two important observations. First, one can observe that the upper and lower bounds are indeed *very tight*, which implies that we can use T_i to give a very accurate measure of the average file downloading time T. Secondly, one can observe that the sojourn times T_i are very close to 1 for $i << K-1$, but when i approaches



(a) $m = 1$ (b) $m = 2$

(c) $m \geq 1$ with FEC

Figure 2.2: Illustration on the last-piece problem: bounds of T_i for $m = 1, 2$ and $m \geq 1$ with FEC. $K = 50$ chunks

$K - 1$, Fig. 2.2(a) (and Fig. 2.2(b)) shows that both bounds approach 2 (approach 1.4) quickly. The increasing downloading time, especially for the last few chunks, depicts the last-piece problem. Intuitively, the reason for this problem is that it becomes more and more difficult for a peer to find other peers which are useful, especially when the peer is very close to finish downloading the whole file. However, one can amend this problem, at least to a certain degree, by simply changing the parameter m. One can observe that when $m = 1$ (as shown in Fig. 2.2(a)), it costs 2 time slots on average to download the last chunk but when $m = 2$ (as shown in Fig. 2.2(b)), it only costs 1.4 time slots to obtain the last chunk. The reason is that when $m = 2$, peers can ask for more peers for bitmaps and thereby increase the chance to find useful peers. Given m, we can derive the bounds of T from Lemma 2.

Theorem 1 *When $m = 1$, the average downloading time $T = K - 2 + H_K + O\left(\frac{\log^2 K}{K}\right)$.*

Proof: In the proof of Lemma 2, we have obtained $T < K - 2 + H_K +$

$O(K^{-1})$. For the lower bound of T, let us denote $A = K - 2 + H_K$, then

$$T = \sum_{i=1}^{K-1} T_i > \sum_{i=1}^{K-1} \frac{1}{1 - \frac{1}{A}\left(\frac{i}{K-i+1}\right)} + O(K^{-1})$$
$$= \frac{A}{A+1}\left(K-1+\frac{K+1}{A+1}\sum_{j=1}^{K-1}\frac{1}{j}\right)+O\left(\frac{\log K}{K}\right) = K-2+H_K+O\left(\frac{\log^2 K}{K}\right).$$

Combining the upper and lower bounds, we know that for any given solution T, there exist two constants α_1 and α_2, so that $K - 2 + H_K + \alpha_1\left(\frac{\log^2 K}{K}\right) \leq T \leq K - 2 + H_K + \alpha_2\left(\frac{\log^2 K}{K}\right)$. Thus Theorem 1 can be shown as claimed. Note that the residual term approach 0 as K increases. ∎

Remark: Note that when $m = 1$, the system corresponds to the "*open and flat*" case of the coupon system [27], in which the authors give an upper bound $T < K + O(\sqrt{K})$. However, the result in Theorem 1 states $T = K - 2 + H_K + O\left(\frac{\log^2 K}{K}\right)$. We know that H_K is the K^{th} harmonic number, $H_K = \log K + \gamma + O(K^{-1})$, where $\gamma = 0.5772...$ is the *Euler-Mascheroni constant*. Thus $T = K + \log K + O(1)$. Therefore, we obtain a tighter bound than [27].

Similarly, we can derive the lower and upper bounds of T from Lemma 2 when $m \geq 2$. Due to the lack of space, we only show the derivation of the upper bound in the following theorem.

Theorem 2 *When $m \geq 2$, the average downloading time $T < K + O\left(\frac{\log K}{K}\right)$.*

Proof:

$$\begin{aligned} T &< \sum_{i=1}^{K-1} \frac{1}{1 - \left[\left(\frac{1}{K-1}\right)\left(\frac{i}{K-i+1}\right)\right]^2} = K - 1 + \sum_{i=1}^{K-1} \frac{\left[\left(\frac{1}{K-1}\right)\left(\frac{i}{K-i+1}\right)\right]^2}{1 - \left[\left(\frac{1}{K-1}\right)\left(\frac{i}{K-i+1}\right)\right]^2} \\ &< K - 1 + \frac{4}{3}\sum_{i=1}^{K-1}\left[\left(\frac{1}{K-1}\right)\left(\frac{i}{K-i+1}\right)\right]^2 \\ &= K - 1 + \frac{4}{3(K-1)^2}\sum_{i=1}^{K-1}\left[\frac{(K+1)^2}{(K-i+1)^2} - \frac{2(K+1)}{K-i+1} + 1\right] \\ &\leq K - 1 + \frac{4}{3}(\zeta(2) - 1) + O\left(\frac{\log K}{K}\right) < K + O\left(\frac{\log K}{K}\right). \end{aligned}$$ ∎

Remark: Since it is necessary to require at least $K - 1$ time slots to finish the downloading of the whole file $\mathcal{F}$, we can conclude by fetching multiple bitmaps (setting $m \geq 2$), the average downloading time is near optimal. To see this, one can compare it with the result in Theorem 1, which states that it takes at least $K + \log(K) + O(1)$ time slots to finish the downloading, and we remove the $\log(K)$ term by getting more than

one bitmap. Setting $m = 2$ is sufficient for achieving the near optimal performance. This result is encouraging and insightful, it shows that due to the diversity of chunks held and the altruistic uploading for every peer, a "simple-design" can achieve very good performance.

Altruistic File Sharing with FEC

We have seen that by fetching bitmaps from multiple peers, the system performance can reach near optimal levels. Here, we provide an *alternative approach* to reach the near optimal performance by using the *forward error correction* (FEC) coding technique [32]. Given a file $\mathcal{F}$, one can encode the original K chunks to $Q = (1+\alpha)K$ chunks with erasure codes before the distribution process. Any peer can reconstruct the original file $\mathcal{F}$ after it receives *any* K distinct chunks of these Q chunks. This technique makes it unnecessary to download the "last" chunk and will ease the last-piece problem, making the system more efficient. To make this claim formally, we have the following theorem:

Theorem 3 *For $m \geq 1$, using FEC with redundancy rate of $\alpha > 0$, the average downloading time $T_{FEC} < K - 2 + (1+\alpha)\log\frac{1+\alpha}{\alpha} + O(K^{-1})$.*

Proof: Note that FEC makes $p_{i,j} = 1 - C_i^j / C_Q^j$ when $1 \leq j \leq i \leq K-1$ and all other equations remain the same. Similarly to the proof of Lemma 2, one can derive that $T_i < \left[1 - \left(\frac{1}{K-1}\right)\left(\frac{i}{Q-i+1}\right)\right]^{-1}$. So

$$\begin{aligned} T_{FEC} &< \sum_{i=1}^{K-1} \frac{1}{1 - \left(\frac{1}{K-1}\right)\left(\frac{i}{Q-i+1}\right)} = \frac{K-1}{K} \sum_{i=1}^{K-1} \left(1 + \frac{1}{\frac{K(Q-i+1)}{Q+1} - 1}\right) \\ &= \frac{(K-1)^2}{K} + \frac{(K-1)(Q+1)}{K^2} \sum_{j=Q-K+1}^{Q-1} \frac{1}{j} + O(K^{-1}) \\ &= K - 2 + (1+\alpha)\log\frac{1+\alpha}{\alpha} + O(K^{-1}). \end{aligned}$$

∎

Remark: Compared with Theorem 1, the harmonic term H_K is replaced with the term $(1+\alpha)\log\frac{1+\alpha}{\alpha}$. Note that, when $\alpha = 0.1$ (i.e. 10% redundancy), this term is less than 2.7. Thus given a particular redundancy rate α, T_{FEC} is bounded by $K-1$ plus a small constant. So by using FEC codes, even if a peer only contacts one other peer for bitmap (i.e. $m = 1$), the average downloading time T can still approach the near optimal value.

Gkantsidis et al. [15] declare that traditional P2P content distribution software as BitTorrent usually suffers from last-piece problem and it could be settled by the network coding technique they propose. In our model we have seen that there exists last-piece problem as Fig. 2.2(a) and Fig. 2.2(b) show. It takes about 2 time slots in average to download the last piece. To illustrate how FEC affects the last-piece problem, let us

consider the upper bound of T_i for a file with $K = 50$ chunks again. By setting $\alpha = 0.1$ (i.e. 10% redundancy), we show the upper bound of T_i in Fig. 2.2(c). This bound holds for all $m \geq 1$. From Fig. 2.2(c), one can observe that the last-piece problem can be eased if we use FEC technique to generate a few redundant chunks. This observation is helpful for the advanced P2P content distribution system design in the future.

2.3 Altruistic File Sharing System with Download and Upload Constraints

In this section, we consider the P2P file sharing system where each peer has a limited bandwidth on the download and upload capacity. Note that this is a more realistic setting than the unlimited upload bandwidth assumption in Section 2.2 and the coupon replication system [27]. This is a very important point since the current Internet, the bottleneck is not at the network core but rather at the edge, and usually the upload capacity of a host is indeed limited (e.g., ADSL or cable system). To simplify our analysis, we only consider the case $m = 1$ (i.e. in each time slot, peer A will first contact *one* other peer randomly in the system to obtain its bitmap). If this peer can help peer A, peer A will request a useful chunk. It is possible that a peer may get multiple requests for chunk. Due to the upload capacity constraint, this peer will only randomly pick one peer to upload. If peer A is chosen, then peer A can download one useful chunk within the current time slot. Otherwise, peer A will remain idle for the current time slot.

2.3.1 Model Formulation

As in Section 2.2, let $p_{i,j}$ denote the probability that a type j peer is useful to the type i peer, $y_i(t)$ denote the number of type i peers in the system at time t. The total number of peers in the system at time t is $y(t) = \sum_{i=1}^{K-1} y_i(t)$. When a type j peer is requested by another peer for its bitmap, the probability that this request comes from a type i peer is $y_i(t)/y(t)$. Thus, the probability that the type j peer is useful to a peer who contacts it is $\beta_j(t) = \sum_{i=1}^{K-1} p_{i,j} y_i(t)/y(t)$.

Assume that peer A contacts peer B and B is of type j. Peer A finds that B is useful and sends B a request for a chunk. Let us consider the probability that A will be chosen by B for service. To derive this probability, we consider how many other peers contacted B for its bitmap. Since there are $y-2$ peers (ignoring A and B) in the system selecting others to contact and B is contacted by a particular peer with probability $1/(y-1)$ (each peer does not contact itself). Thus the number of peers that contacted B, denoted by the random variable R, is the number of successes in a sequence of $y-2$ independent Bernoulli trials, or $R \sim$

Bernoulli$(y-2, \frac{1}{y-1})$. Since $y-2$ is large and $(y-2)/(y-1) \sim 1$, R can be approximated as a Poisson random variable with mean 1, thus R has a probability mass function of $f_R(k) = e^{-1}/k!$, for $k \in \{0, 1, \ldots\}$.

Assume $R = r$ (i.e. peer B was contacted by r peers for its bitmap). The probability that peer B is useful to a peer in R is $\beta_j(t)$. Thus B receives k requests for chunk with probability $C_r^k \beta_j^k(t)(1-\beta_j(t))^{r-k}$ for $k \leq r$. When A contacts B, finds B is useful and also sends B a request for chunk, the probability that A is chosen by B for service is

$$\alpha_{j,r}(t) = \sum_{k=0}^{r} C_r^k \beta_j^k(t)(1-\beta_j(t))^{r-k} \frac{1}{k+1} = \frac{1-(1-\beta_j(t))^{r+1}}{(r+1)\beta_j(t)}.$$

The system can be modeled as a Markov process $\boldsymbol{Y}(t) = \{y_i(t)\}$. Again, it is easy to verify that $(\boldsymbol{Y}(t))_{t\geq 0}$ is a Markov process taking its values in $\mathcal{Z}_+^{K-1}$. The non-zero transition rates of this Markov process, for all $i \in \{1, \ldots, K-1\}$ is

$$\begin{aligned}
\boldsymbol{Y} &\longrightarrow \boldsymbol{Y} + e_1 && \text{with rate } \lambda, \\
\boldsymbol{Y} &\longrightarrow \boldsymbol{Y} - e_i + e_{i+1} && \text{with rate } y_i \sum_{j=1}^{K-1} \left[\frac{y_j}{y} p_{i,j} \sum_{r=0}^{\infty} \frac{e^{-1}}{r!} \alpha_{j,r} \right] \\
& && i \in \{1, \ldots, K-2\} \\
\boldsymbol{Y} &\longrightarrow \boldsymbol{Y} - e_{K-1} && \text{with rate } y_{K-1} \sum_{j=1}^{K-1} \left[\frac{y_j}{y} p_{K-1,j} \sum_{r=0}^{\infty} \frac{e^{-1}}{r!} \alpha_{j,r} \right].
\end{aligned}$$

For a large population asymptotic regime, this density dependent jump Markov process converges to the solution of the system of differential equations

$$\begin{aligned}
\frac{dy_1(t)}{dt} &= \lambda - y_1(t) \sum_{j=1}^{K-1} \left[\frac{y_j(t)}{y(t)} p_{1,j} \sum_{r=0}^{\infty} \frac{e^{-1}}{r!} \alpha_{j,r}(t) \right], \\
\frac{dy_i(t)}{dt} &= y_{i-1}(t) \sum_{j=1}^{K-1} \left[\frac{y_j(t)}{y(t)} p_{i-1,j} \sum_{r=0}^{\infty} \frac{e^{-1}}{r!} \alpha_{j,r}(t) \right] - \\
&\quad y_i(t) \sum_{j=1}^{K-1} \left[\frac{y_j(t)}{y(t)} p_{i,j} \sum_{r=0}^{\infty} \frac{e^{-1}}{r!} \alpha_{j,r}(t) \right], \qquad i = 2, \ldots, K-1.
\end{aligned}$$

with some initial condition $\boldsymbol{Y}(0)$.

2.3.2 Steady State Analysis

We focus on the steady state performance and we are interested in its *equilibrium point*. In other words, the operating point wherein $dy_i/dt = 0$ for $1 \leq i \leq K-1$. Define T_i as the sojourn time for type i peer. It follows from Little's theorem that $\lambda T_i = y_i$. Let the average file downloading

time be $T = \sum_{j=1}^{K-1} T_j$, one can obtain the following equations at the equilibrium point:

$$\frac{1}{T_i} = \sum_{j=1}^{K-1} \Big(\frac{T_j}{T} p_{i,j} \sum_{r=0}^{\infty} \frac{e^{-1}}{r!} \alpha_{j,r} \Big), \quad i = 1, 2, ..., K-1 \qquad (2.4)$$

where

$$\alpha_{j,r} = \frac{1-(1-\beta_j)^{r+1}}{(r+1)\beta_j} \quad \text{and} \quad \beta_j = \sum_{i=1}^{K-1} \frac{T_i}{T} p_{i,j}, \quad j = 1, 2, ..., K-1.$$

In Section 2.2, we have shown that a P2P file sharing system that has only download capacity constraint is very efficient. With both download and upload capacity constraints, the performance of the system will not be as good. In this section, we seek to derive the bounds of T_i (and thereby T) to gain insight on how the upload capacity constraint can affect the system performance. Let us first state the upper and lower bounds of the sojourn time T_i.

Theorem 4 *The sojourn times T_i satisfies*

$$\frac{1}{1-e^{-1}} + O\left(\frac{\log K}{K}\right) < T_i < \left[\frac{1}{1-e^{-1}}\right] \left[\frac{1}{1-\left(\frac{1}{K-1}\right)\left(\frac{i}{K-i+1}\right)}\right].$$

Proof: Because $\beta_j < 1$, $r \geq 0$, we have $\alpha_{j,r} \geq 1/(r+1)$. From Eq. (2.4), we use the same technique in proofing the lower bound of q_i in Lemma 2:

$$\begin{aligned} \frac{1}{T_i} &\geq \sum_{j=1}^{K-1} \left(\frac{T_j}{T} p_{i,j} \sum_{r=0}^{\infty} \frac{e^{-1}}{r!} \frac{1}{r+1} \right) = (1-e^{-1}) \sum_{j=1}^{K-1} \frac{T_j}{T} p_{i,j} \\ &> \left[1-e^{-1}\right] \left[1 - \left(\frac{1}{K-1}\right)\left(\frac{i}{K-i+1}\right)\right]. \end{aligned}$$

Therefore, the upper bound of T_i is obtained. For the lower bound of T_i, we have $\alpha_{j,r} \leq [1+r(1-\beta_j)]/(r+1)$ because $\beta_j < 1$ and $r \geq 0$. Thus

$$\frac{1}{T_i} \leq \sum_{j=1}^{K-1} \left[\frac{T_j}{T} \sum_{r=0}^{\infty} \frac{e^{-1}}{r!} \frac{1+r(1-\beta_j)}{r+1} \right] = 1 - e^{-1} \sum_{j=1}^{K-1} \frac{T_j}{T} \beta_j < 1 - \frac{e^{-1}}{K-1} \sum_{j=1}^{K-1} \beta_j.$$

One can obtain an upper bound on the summation term as

$$\begin{aligned} \sum_{j=1}^{K-1} \beta_j &= \sum_{i=1}^{K-1} \frac{T_i}{T} \left(K - 1 - \frac{i}{K-i+1} \right) = K - \frac{K+1}{T} \sum_{i=1}^{K-1} \frac{T_i}{K-i+1} \\ &> K - \frac{K+1}{K-1} \sum_{i=1}^{K-1} \frac{1}{(K-i+1)(1-e^{-1})\left[1 - \left(\frac{1}{K-1}\right)\left(\frac{i}{K-i+1}\right)\right]} \\ &= K - \frac{H_K}{1-e^{-1}} + O\left(\frac{\log K}{K}\right). \end{aligned}$$

Finally, the lower bound of T_i can be obtained as

$$\frac{1}{T_i} < 1 - \frac{e^{-1}}{K-1} \sum_{j=1}^{K-1} \beta_j < 1 - e^{-1} + O\left(\frac{\log K}{K}\right). \quad \blacksquare$$

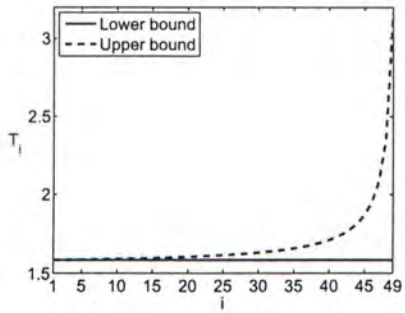


Figure 2.3: Numerical results illustrated for the bounds of T_i for $m = 1$ when $K = 50$

Figure 2.3 illustrates the upper and lower bounds of T_i for a file with $K = 50$ chunks and $m = 1$. Notice that the lower bound of T_i is rather loose since it is not related to the index i. Nevertheless, the spread of the bounds is tight for most values of T_i. Another observation is that for small values of i, T_i is not close to 1 any more as in the case of Section 2.2, but rather, close to $1/(1 - e^{-1})$. This performance degradation is contributed by the constraint on the upload capacity. In other words, if one limits the number of chunks that a peer can upload each time slot, it takes longer, on average, to obtain the file. Lastly, with the upper and lower bounds of T_i, one can derive the average downloading time T.

Theorem 5 *The average downloading time T satisfies*

$$\frac{K}{1-e^{-1}} + O(\log K) < T < \frac{1}{1-e^{-1}}(K - 2 + H_K) + O(K^{-1})$$

Proof: Given the upper bound of T_i, one can use the approach similar to Lemma 2 to derive that $T = \sum_{i=1}^{K-1} T_i < (K-2+H_K)/(1-e^{-1}) + O(K^{-1})$. With the lower bound of T_i, we have

$$T = \sum_{i=1}^{K-1} T_i > (K-1)\left[\frac{1}{1-e^{-1}} + O\left(\frac{\log K}{K}\right)\right] = \frac{K}{1-e^{-1}} + O(\log K). \blacksquare$$

Compared with Theorem 1, the average downloading time has been scaled up by a factor of $1/(1 - e^{-1})$ when K is large. It is interesting to explore whether using FEC can improve the performance of the system. We have the following result.

Lemma 3 *When one uses FEC in this system, the bounds of T_i as specified in Theorem 4 and the average downloading time T as specified in Theorem 5 will remain the same.*

Proof: Similar to Section 2.2.2, FEC will increase the value of $p_{i,j}$ and other equations remain the same. Thus the upper bound of T_i in Theorem 4 still hold. Notice that we just replaced $p_{i,j}$ by 1 in the proof of the lower bound of T_i in Theorem 4. And $p_{i,j} \leq 1$ still holds even with FEC, thus the lower bound of T_i in Theorem 4 still holds too. We know that Theorem 5 is derived from 4 directly, thus the bounds in Theorem 5 also remain the same. ∎

Lemma 3 implies that FEC could not improve the performance very much. It can be explained as follows. The random peer selection policy may cause request collision since a peer may receive multiple chunk requests but can only serve one peer. Other peers requesting chunk from the same peer will waste their time slot.

2.4 Incentive File Sharing via Coordinated Matching

From Theorem 5, one can observe that when there are both upload / download capacity constraints on cooperative peers and peers use a random peer selection policy, the average downloading time $T = \frac{K}{1-e^{-1}} + O(\log K)$, where the coefficient of the term K is $\frac{1}{1-e^{-1}} \approx 1.58$. The system performance degrades as compared with the file sharing system without upload capacity constraint where the coefficient of term K is 1. The performance degradation can be explained as follows: the random peer selection may cause request collision since a peer may receive multiple chunk requests but can only serve one request. Therefore, some peers may waste the download opportunity and remain idle for a time slot. For the case of unlimited upload capacity, all requests can be satisfied, hence, the performance is better.

One may ask, in the system with both download and upload capacity constraints, can the system still achieve good performance by using peer selection algorithms other than the random policy? In the following, we show that by running a maximal matching algorithm (usually regard as an "easy problem" with efficient polynomial algorithm) at the beginning of every time slot, one can significantly improve the system performance. Also, we show that with built-in incentive mechanisms, this approach can also provide very good performance.

2.4.1 Without Incentive Mechanism

We assume at the beginning of each time slot, every peer will run some distributed *maximal matching algorithm* [17], or gets the help from some

central server, so that peer A will find peer B as its neighbor while peer B will also find A as its neighbor. If the matching process is *independent* of the chunks held by each peer, then given peer A, the probability that peer B is of type i is y_i/y where y_i is the number of type i peers and y is the total number of peers in the system. At the current time slot, peer A can only communicate with peer B and vice versa and the matched peers can upload and download at most one chunk per time slot.

Let us first study the system without incentive mechanism. When peer A and peer B are matched, peer A will help peer B if and only if peer A is useful to peer B (i.e. $\mathcal{F}_A \setminus \mathcal{F}_B \neq \emptyset$); similarly peer B will help peer A if and only if $\mathcal{F}_B \setminus \mathcal{F}_A \neq \emptyset$. Since the selection of neighbor is independent of peers' type, we get the differential equations for the number of type i peers as

$$\frac{dy_i(t)}{dt} = \begin{cases} \lambda - y_1 \sum_{j=1}^{K-1} \frac{y_j(t)}{y(t)} p_{1,j} & i = 1 \\ y_{i-1}(t) \sum_{j=1}^{K-1} \frac{y_j(t)}{y(t)} p_{i-1,j} - y_i(t) \sum_{j=1}^{K-1} \frac{y_j(t)}{y(t)} p_{i,j} & i = 2, \ldots, K-1. \end{cases} \tag{2.5}$$

One can find that, Eq. (2.5) is equivalent to the differential equations given in Eq. (2.2) where peers have unlimited upload capacity and $m = 1$. Thus, the asymptotic bounds given in Theorem 1 still holds for this model, which implies $T = K + \log(K) + O(1)$

Remark: Both the download and upload capacity are one chunk per time slot, each peer has the same constraints as that in Section 2.3. However, we have better performance when matching is used instead of the random peer selection. The random peer selection may cause request collision (i.e. a peer may receive multiple chunk requests but it can only serve one request due to its upload capacity), so the download capacities of the unserved peers are wasted. But if peers are matched at the beginning of each time slot, then the performance is greatly improved, approaching the performance of the random peer selection with *unlimited* upload capacity.

2.4.2 With Incentive Mechanism

Let us study the system with coordinated matching but with an incentive mechanism. Namely, given a pair of neighboring peers: peer A and peer B, both of them will perform chunk transfer *iff* both of them are useful to each other (i.e., $\mathcal{F}_A \setminus \mathcal{F}_B \neq \emptyset$ and $\mathcal{F}_B \setminus \mathcal{F}_A \neq \emptyset$). In this case, peer A and B will obtain one new chunk from each other in the current time slot. We use this model to capture the "*tit-for-tat*" incentive mechanism in the BT protocol. With this mechanism, the probability that a type i peer can exchange chunk with a type j peer is

$$p'_{i,j} = \begin{cases} 1 - \frac{C_i^j}{C_K^j} & 1 \leq j \leq i \leq K-1, \\ 1 - \frac{C_j^i}{C_K^i} & 1 \leq i < j \leq K-1. \end{cases} \tag{2.6}$$

Let us first state some important properties of $p'_{i,j}$.

Lemma 4 *$p'_{i,j}$ has the following properties: (1) $p'_{i,j} = p'_{j,i}$; (2) $p'_{i,j} = p'_{K-j,K-i}$ and (3) $p'_{i,j}$ is an increasing function of j when $j \leq i$, and $p'_{i,j}$ is a decreasing function of j when $j \geq i$.*

Proof: The proof of property (1) is trivial. To prove property (2), we consider the following three cases:

- **Case 1:** $1 \leq j \leq i$: we have $p'_{i,j} = 1 - C_i^j/C_K^j = 1 - C_{K-j}^{K-i}/C_K^i$. $j \leq i$ implies $K-i \leq K-j$, therefore, $p'_{K-j,K-i} = 1 - C_{K-j}^{K-i}/C_K^{K-i} = 1 - C_{K-j}^{K-i}/C_K^i$. So we get $p'_{i,j} = p'_{K-j,K-i}$
- **Case 2:** $i < j \leq K-1$: We have $p'_{i,j} = p'_{j,i} = p'_{K-i,K-j} = p'_{K-j,K-i}$.

To prove property (3), let us consider the following cases:

- **Case 1:** $1 \leq j' < j \leq i \leq K-1$:

$$p'_{i,j} - p'_{i,j'} = \left(1 - \frac{C_i^j}{C_K^j}\right) - \left(1 - \frac{C_i^{j'}}{C_K^{j'}}\right) = \left(1 - \frac{C_{K-j}^{K-i}}{C_K^i}\right) - \left(1 - \frac{C_{K-j'}^{K-i}}{C_K^i}\right) > 0$$

- **Case 2:** $i \leq j' < j \leq K-1$: Since $K-j < K-j' \leq K-i$, we have

$$p'_{i,j} - p'_{i,j'} = p'_{K-i,K-j} - p'_{K-i,K-j'} < 0. \qquad \blacksquare$$

To simplify our notation, let us denote $w_{i,j} = p'_{i,j} + p'_{i,K-j}$ $(i,j = 1,\ldots,K-1)$. It is easy to show that $w_{i,j} = w_{i,K-j} = w_{K-i,j} = w_{j,i}$.

Lemma 5 *For a given i, $w_{i,j}$ is an increasing (or decreasing) function of j for $j \leq K/2$ (for $j \geq K/2$).*

Proof: Consider $i \leq K/2$ first, in this case,

(1) $j \leq i$, we have

$$\begin{aligned} w_{i,j} - w_{i,j-1} &= p'_{i,j} + p'_{i,K-j} - (p'_{i,j-1} + p'_{i,K-j+1}) \\ &= (p'_{i,j} - p'_{i,j-1}) + (p'_{i,K-j} - p'_{i,K-j+1}) > 0. \end{aligned}$$

(2) $i < j \leq K/2$, we have

$$\begin{aligned} w_{i,j} - w_{i,j-1} &= \left(1 - \frac{C_j^i}{C_K^i} + 1 - \frac{C_{K-j}^i}{C_K^i}\right) - \left(1 - \frac{C_{j-1}^i}{C_K^i} + 1 - \frac{C_{K-j+1}^i}{C_K^i}\right) \\ &= \frac{1}{C_K^i}\left[C_{K-j}^{i-1} - C_{j-1}^{i-1}\right] > 0. \end{aligned}$$

Combine case (1) and (2), we know when $i \leq K/2$, $w_{i,j}$ is increasing if $j \leq K/2$. Since $w_{i,j} = w_{i,K-j}$, $w_{i,j}$ is decreasing if $j \geq K/2$. Because $w_{i,j} = w_{K-i,j}$, the above results hold for $i > K/2$. $\blacksquare$

Lemma 6 $T_i = T_{K-i}$.

Proof: We take a reverse view in the steady state so that (1) we regard the departure as arrival; (2) if peer A's storage is $\mathcal{F}_A$, we just imagine there is no peer A but its complementary peer $\bar{A}$ with storage $\mathcal{F}_{\bar{A}} = \mathcal{F} \setminus \mathcal{F}_A$. So originally T_i is the average time for peer A to stay in type i(i.e. with i chunks), but now the average time for peer $\bar{A}$ to stay in type $(K-i)$: $T'_i = T_{K-i}$. From Lemma 4 we know $p'_{i,j} = p'_{K-i,K-j}$. So the "reversed system" is identical to the original system which implies $T'_i = T_i$. Thus we get $T_i = T_{K-i}$. ∎

Similar to the steady state analysis in previous section, we have the equations for T_i:

$$\frac{1}{T_i} = \sum_{j=1}^{K-1} \frac{T_j}{T} p'_{i,j}, \qquad i = 1, 2, \ldots, K-1, \tag{2.7}$$

where $T = \sum_{i=1}^{K-1} T_i$.

Lemma 7 *For $i \leq K/2$, T_i is a decreasing sequence: $2 > T_1 > T_2 > \ldots > T_{\lfloor K/2 \rfloor}$.*

Proof: Let $1 \leq i' < i \leq \lfloor K/2 \rfloor$. Based on Lemma 6, we have

$$\frac{1}{T_i} = \sum_{j=1}^{K-1} \frac{T_j}{T} p'_{i,j} = \frac{1}{2} \sum_{j=1}^{K-1} \frac{T_j}{T} (p'_{i,j} + p'_{i,K-j}) = \frac{1}{2} \sum_{j=1}^{K-1} \frac{T_j}{T} w_{i,j}.$$

Similarly, $\frac{1}{T_{i'}} = \frac{1}{2} \sum_{j=1}^{K-1} \frac{T_j}{T} w_{i',j}$. From Lemma 5 and $w_{i,j} = w_{j,i}$, we have

$$\frac{1}{T_i} - \frac{1}{T_{i'}} = \sum_{j=1}^{K-1} \frac{T_j}{T} (w_{i,j} - w_{i',j}) = \sum_{j=1}^{K-1} \frac{T_j}{T} (w_{j,i} - w_{j,i'}) > 0.$$

Thus $T_i < T_{i'}$, and the upper bound of T_1 is

$$T_1 = \frac{2}{\sum_{j=1}^{K-1} \frac{T_j}{T} w_{1,j}} < \frac{2}{\sum_{j=1}^{K-1} \frac{T_j}{T} w_{1,1}} = \frac{2}{w_{1,1}} = 2. \qquad ∎$$

Theorem 6 *Using the incentive mechanism stated above, the bounds on the average downloading time T are*

$$K - 4 + 2H_K + O\left(\frac{\log K}{K}\right) \leq T \leq K - 2 + 4H_K + O\left(\frac{\log K}{K}\right).$$

Proof: Base on Lemma 5, Lemma 6 and Lemma 7, we have

$$\frac{1}{T_i} = \sum_{j=1}^{K-1} \frac{T_j}{T} p'_{i,j} = \frac{1}{2} \sum_{j=1}^{K-1} \frac{T_j}{T} w_{i,j} \leq \frac{1}{K-1} \sum_{j=1}^{K-1} \frac{T_j}{T} \cdot \sum_{j=1}^{K-1} p'_{i,j} = \frac{1}{K-1} \sum_{j=1}^{K-1} p'_{i,j}$$

$$= \frac{1}{K-1}\left[\sum_{j=1}^{i}\left(1-\frac{C_i^j}{C_K^j}\right)+\sum_{j=i+1}^{K-1}\left(1-\frac{C_j^i}{C_K^i}\right)\right]$$
$$= 1-\frac{1}{K-1}\left[\frac{i}{K-i+1}+\frac{K-i}{i+1}-\frac{1}{C_K^i}\right].$$

Therefore, we obtain the lower bound of T as

$$T = \sum_{i=1}^{K-1} T_i \geq \sum_{i=1}^{K-1} \frac{1}{1-\frac{1}{K-1}\left[\frac{i}{K-i+1}+\frac{K-i}{i+1}-\frac{1}{C_K^i}\right]} > \sum_{i=1}^{K-1} \frac{1}{1-\frac{1}{K-1}\left[\frac{i}{K-i+1}+\frac{K-i}{i+1}-1\right]}$$
$$= \sum_{i=1}^{K-1}\left[\frac{K-1}{K+2}+\frac{K^2-1}{K^2+2K}\left(\frac{1}{i}+\frac{1}{K-i}\right)\right] = K-4+2H_K+O\left(\frac{\log K}{K}\right).$$

According to Eq. (2.7) and Lemma 7, we have

$$\frac{1}{T_i} = \sum_{j=1}^{i}\frac{T_j}{T}\left(1-\frac{C_i^j}{C_K^j}\right)+\sum_{j=i+1}^{K-1}\frac{T_j}{T}\left(1-\frac{C_j^i}{C_K^i}\right) = 1-\left(\sum_{j=1}^{i}\frac{T_j}{T}\frac{C_i^j}{C_K^j}+\sum_{j=i+1}^{K-1}\frac{T_j}{T}\frac{C_j^i}{C_K^i}\right)$$
$$> 1-\frac{T_1}{T}\left(\sum_{j=1}^{i}\frac{C_i^j}{C_K^j}+\sum_{j=i+1}^{K-1}\frac{C_j^i}{C_K^i}\right) > 1-\frac{2}{K-1}\left(\frac{i}{K-i+1}+\frac{K-i}{i+1}\right).$$

Thus for $i = 3 \ldots K-3$ (assuming $K \geq 5$), we have

$$T_i < \frac{1}{1-\frac{2}{K-1}\left(\frac{i}{K-i+1}+\frac{K-i}{i+1}\right)} = \frac{(K-1)(K-i+1)(i+1)}{K^2i-K^2-Ki^2+3Ki-3i^2-2K-1}$$
$$< \frac{(K-1)(K-i+1)(i+1)}{K^2i-K^2-Ki^2+K} = \frac{K-1}{K}+\frac{2(K-1)}{K-2}\left(\frac{1}{K-i-1}+\frac{1}{i-1}\right).$$

Thus the upper bound of T is

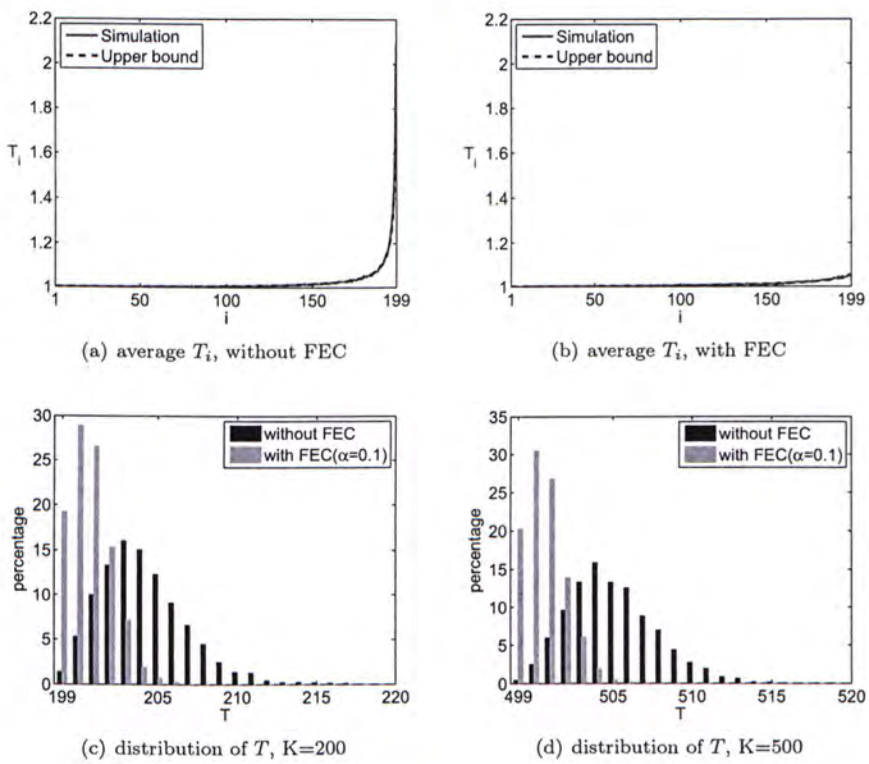
$$T = \sum_{i=1}^{K-1} T_i < 4T_1 + \sum_{i=3}^{K-3}\left[\frac{K-1}{K}+\frac{2(K-1)}{K-2}\left(\frac{1}{K-i-1}+\frac{1}{i-1}\right)\right]$$
$$< 8+\frac{(K-1)(K-5)}{K}+\frac{4(K-1)}{K-2}(H_{K-4}-1)+O(K^{-1})$$
$$= K+4H_K-2+O\left(\frac{\log K}{K}\right). \qquad \blacksquare$$

Remark: Given the upper and lower bounds in Theorem 6, one can conclude that when incentive mechanism is employed to enhance fairness, the performance of the file sharing system still achieves better than the random peer selection policy in Section 5 wherein no fairness is guaranteed and free riders can benefit from peers' altruistic service. Therefore, it is important for a system to help peers avoid waste of download capacity (request collision). Under the assistance of peer matching mechanism (such as coordinated matching presented) even if the uploading and download capacity is tightly constrained, the system can still provide good performance with a fairness guarantee.

2.5 Simulation

In this section, we carry out simulations to (1) validate our analytical results and (2) obtain other performance measures such as probability distribution of file downloading time. Unless we state otherwise, the arrival process of peers is a Poisson process with $\lambda = 2.0$. Since the system is slotted, peers arrive at time slot t will obtain the initial chunk and will start participating in the file sharing process in the beginning of time slot $t + 1$. The file that will be shared by all peers has $K = 200$ chunks. We also have results for $K = 500$, but due to the lack of space we mainly discuss the case $K = 200$.

Experiment 1: The goal of this experiment is to validate the analytical results in Section 2.2 and to illustrate the *probability density function* of the file download time. For this experiment, we set $m=1$ or equivalently, this corresponds to the coupon model [27].



(a) average T_i, without FEC

(b) average T_i, with FEC

(c) distribution of T, K=200

(d) distribution of T, K=500

Figure 2.4: T_i and T for $m = 1$, constraint on download capacity only

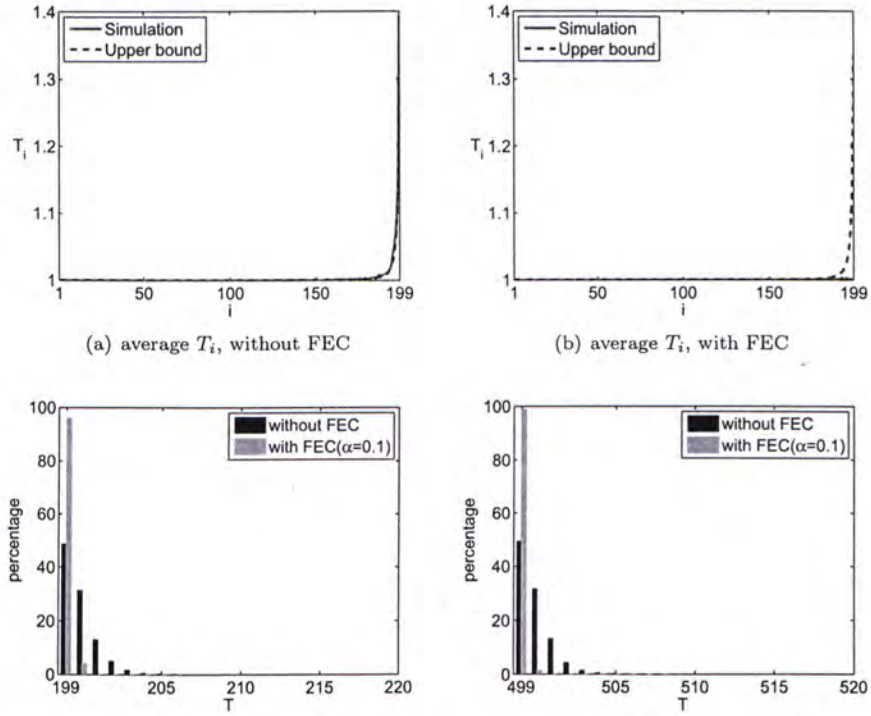
Fig. 2.4(a) presents the average sojourn time T_i for a file with $K = 200$ chunks. We compare the simulation results and the analytical results[2]. This indicates that our analytical result is very accurate. Fig. 2.4(b)

[2]For the analytical results, since the spread of the bounds is very tight, we simply plot the upper bound of T_i

illustrates T_i under similar setting but we enable the FEC with 10% redundancy (i.e., $\alpha = 0.1$). One can conclude that the analytical model is again very accurate and that using FEC can resolve the last-piece problem. Fig. 2.4(c)-(d) illustrate the probability density function for the average file downloading time T, with and without using FEC, for $K = 200$ and 500 respectively. When $K = 200$ ($K = 500$) without using FEC, the analytical average file downloading time is $T = 203.88$ ($T = 504.79$), and the simulation average file downloading time is $T = 204.11$ ($T = 504.99$). When $K = 200$ ($K = 500$) and FEC is enabled, the analytical average file downloading time is $T = 200.64$ ($T = 500.64$) while simulation average file downloading time is $T = 200.72$ ($T = 500.64$). We can observe that by using FEC, not only one can reduce the average T but also the variance of T.

Experiment 2: This experiment is to validate the results in Section 2.2 when $m > 1$. According to our analysis, there is not much difference between $m = 2$ and $m > 2$ since the average downloading time T will be bounded by K. For this experiment, we set $m = 2$. Fig.2.5(a) presents the average sojourn time T_i without FEC. The simulation results are similar to the analytical results again. Comparing Fig. 2.5(a) and Fig. 2.4(a), one can find that last-piece problem is not so severe for $m = 2$. T_i raises only for the last five chunks. If we deploy FEC ($\alpha = 0.1$) together with $m = 2$, the last-piece problem can be resolved and this is illustrated in Fig. 2.5(b). Notice that we only give out a loose upper bound of T_i in Fig. 2.5(b), which is also the upper bound of the system without FEC in Fig. 2.5(a). Now we examine the probability density function of T in Fig. 2.5(c). Without FEC, 50% peers finished in $K - 1$ time slots and 80% peers finished in less than or equal to K time slots. After we enable the FEC with $\alpha = 0.1$, 96% peers finished in $K - 1$ time slots and all finished in less than or equal to K time slots. One can conclude that the average downloading time T is close to the optimal value of 199 (or $K-1$), and the variance of T is also reduced. When $K = 200$ ($K = 500$) and without FEC, our analysis gives an upper bound of the average downloading time $T \leq 200$ ($T \leq 500$), and the simulation is $T = 199.83$ ($T = 499.78$). After using FEC with $\alpha = 0.1$, the analytical upper bound of T still holds, while the simulation gives $T = 199.04$ ($T = 499.01$).

Experiment 3: This experiment is to validate the altruistic system with download and upload capacity constraints in Section 2.3. We consider $m=1$ in our analysis thus we set $m = 1$ in this simulation. Fig.2.6(a) presents the average sojourn time T_i without FEC. The simulation results and the analytical results match very well, i.e. our theoretical upper bound is very tight. Comparing Fig. 2.6(a) and Fig. 2.6(b), we observe that FEC eases the last-piece problem, but most of T_i remain the same and they cannot approach to 1 even with FEC. The reason is that the performance degradation is due to the request collision but not the last-piece



(a) average T_i, without FEC

(b) average T_i, with FEC

(c) distribution of T, K=200

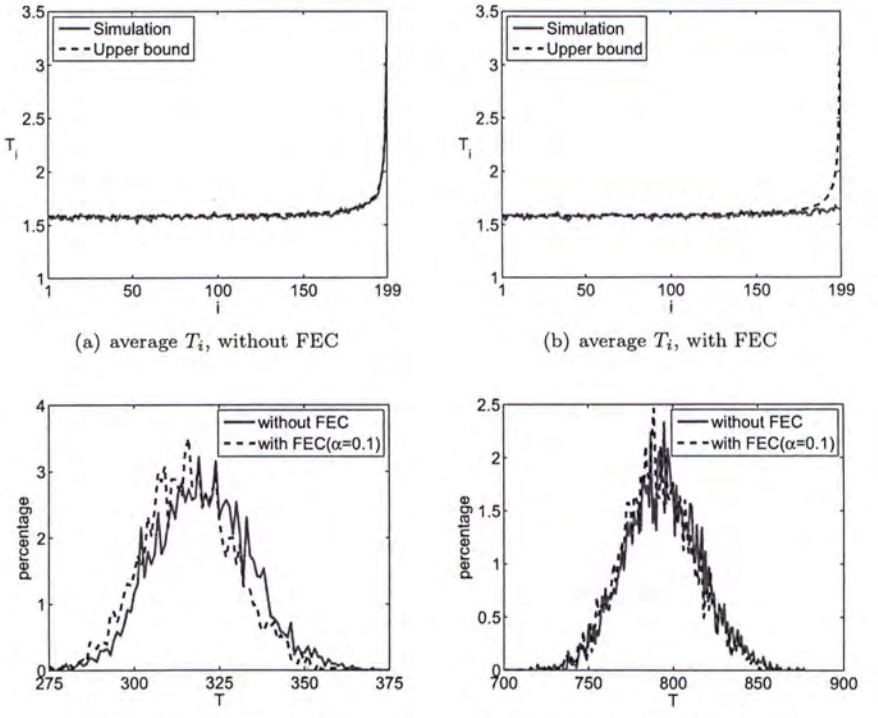
(d) distribution of T, K=500

Figure 2.5: T_i and T for $m = 2$, constraint on download capacity only

problem. Also note that when we have upload and download capacity constraints, the variance on T is significantly larger than the previous experiments. This can by confirmed by Fig. 2.6(c), the downloading time T varies in a wide range, from 275 to 375, and using FEC does not reduce the variance very much. When $K = 200$ ($K = 500$) and without FEC, our analytical bound of the average downloading time is $T \leq 322.53$ ($T \leq 798.56$), while the simulation gives $T = 319.99$ ($T = 793.67$). With FEC, the upper bound still holds, and the simulation result is $T = 316.06$ ($T = 791.01$), these show that using FEC in this type of system cannot improve T very much.

Experiment 4: This experiment is to validate the coordinated matching system with incentive mechanism as described in Section 5. Fig.2.7(a) presents the average sojourn time T_i without FEC. One can observe that the gap between the simulation results and our analytical upper bound is small. Also, one can observe both the *last-piece problem* and *first-piece problem*[3] in our analytical bound and simulation result. The *first-piece problem* can be explained as follows. When a peer has very few chunks, it can hardly help other peers. Due to the incentive mechanism, it is

[3]This problem is reported as first block problem in [23] by measurement study as the slow startup due to choking.



(a) average T_i, without FEC

(b) average T_i, with FEC

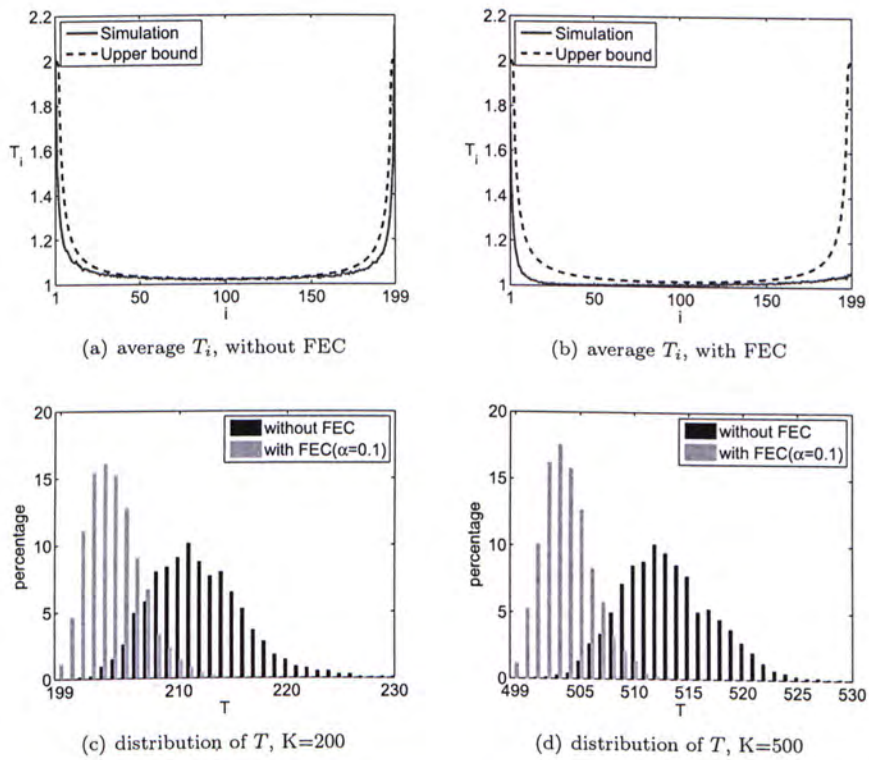
(c) distribution of T, K=200

(d) distribution of T, K=500

Figure 2.6: T_i and T for $m = 1$, constraint on upload and download capacity

difficult for this peer to obtain service from others. We can observe that FEC does well in easing the last-piece problem, but is not so good at easing the *first-piece problem*, as Fig. 2.7(b) has indicated. From Fig. 2.7(c) and 2.7(d), one can observe that the average and variance of file downloading time can be reduced when FEC is deployed. Another important observation is that when FEC is deployed, the performance measures of T (both for the average and variance) are significantly improved as compared with the results in Experiment 3 wherein both systems are under the upload and download capacity constraints. When $K = 200$ ($K = 500$) and without FEC, our analysis gives an upper bound of the average downloading time $T \leq 221.50$ ($T \leq 525.17$), and the simulation is $T = 211.78$ ($T = 513.10$). After using he FEC with $\alpha = 0.1$, the analytical upper bound of T still holds, and the simulation gives $T = 203.90$ ($T = 503.77$). This validates our analytical models.

□ **End of chapter.**



(a) average T_i, without FEC

(b) average T_i, with FEC

(c) distribution of T, K=200

(d) distribution of T, K=500

Figure 2.7: T_i and T for coordinated matching, incentive mechanism, with constraint on upload and download capacity

Chapter 3

An ISP-friendly Protocol

3.1 Simple Mathematical Models

We consider a P2P file distribution system which disseminates files to a large number of peers. The file to be disseminated, say $\mathcal{F}$, is divided into many pieces. Formally, we have $\mathcal{F} = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_K\}$ in which the file $\mathcal{F}$ has $K \geq 1$ pieces, $\mathcal{C}_i$ is the i^{th} piece of $\mathcal{F}$ and $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ for $i \neq j$. A peer that holds all pieces of the file is called a *seeder* while a peer that holds a subset of pieces is called a *leecher*. To download the file, a peer (or leecher) needs to download all K pieces.

Before we present the analysis of an ISP-friendly protocol, let us consider the current P2P file distribution system, such as BitTorrent, in which peers do not consider the boundary between ISPs in their data transfer. We call such kind of P2P file distribution as "*random downloading*". What we are interested in is the amount of cross-ISP traffic. Assume that the number of peers in the P2P system is N, n of which are within the ISP a. Considering a randomly chosen peer which resides in the ISP a, the probability of choosing a peer outside ISP a for the data transfer is

$$f = 1 - \frac{n}{N}. \tag{3.1}$$

Thus the expected fraction of file content which is downloaded from(upload to) peers outside ISP a is $(1 - n/N)$. The total amount of incoming(outgoing) cross-ISP traffic is approximately $(n(1 - \frac{n}{N}) * \text{file size})$. This represents a large volume of cross-ISP traffic because usually there are many peers in a P2P file distribution system, for instance, N is much larger than n and n is relatively large.

In analyzing the performance of an ISP-friendly protocol, we seek to derive the amount of cross-ISP traffic if peers are willing to follow the *exploiting-the-locality principle* (ELP). Obviously, only when the reduction of cross-ISP traffic is high, then one should consider designing and implementing an ISP-friendly file distribution protocol. In our analysis, we concentrate on two common scenarios in P2P file distribution: regular

peer arrival and a big bursty peer arrival(flash crowd).

3.1.1 Assumptions

Unlike previous work which focused on the performance modeling of file downloading time, we model the amount of cross-ISP traffic. For our mathematical model, we make the following assumptions:

- Peer arrival process is characterized by a Poisson process with an average rate λ.
- Peers are all *persistent* in the sense that they will not abort before they finish the file download.
- To ensure file availability, we assume there exists at least one *seeder* in the system: some peers are willing to publish the original file to the P2P network.
- Whenever a peer (or leecher) obtains all pieces of a file, the peer will leave the system immediately.
- The piece diversity of the P2P system is very good so that peers will be interested in each other with high probability.

Note that the last assumption is a common assumption for most fluid models of P2P systems [12,31]. One may argue that the *last piece problem* may destroy this assumption, but the measurement results in [1,23] and the stochastic analysis [24] show that peers show interest of each other most of the time and the last piece problem only affects the last few pieces. Its effect in the mathematical model can be safely ignored for a large file which contains thousands of pieces. This assumption means that the downloading rate for a given peer can be represented by a random variable which is independent of its downloading progress.

Note that based on the ELP, if there exists a seeder in an ISP, then all peers in that ISP will never download pieces from external peers and the incoming cross-ISP traffic is zero. This is a trivial case. We consider a more interesting case wherein the seeder does not reside in an ISP. The derivation of the cross-ISP traffic for an ISP-friendly protocol is complicated and it depends on the specific implementation of the protocol, but instead, one can derive a upper and lower bound of this measure. Before we present the formal analysis, let us use an example as shown in Figure 3.1 to illustrate the idea. The file has 20 pieces and at this moment, there are 3 peers within the ISP. Let v_i be the fraction of progress in the file download for peer i. In this example, we have $v_1 = 0.3$ (6 pieces), $v_2 = 0.15$ (3 pieces) and $v_3 = 0.2$ (4 pieces). Since peers follows the ELP, only those missing pieces by all peers would be downloaded through the cross-ISP link. How many pieces would be downloaded through the cross-ISP link before the next peer departure? In the best case, when all peers

possess *different* pieces from each others, then the external download will be $d = 1 - \sum_{i=1}^{3} v_i = 0.35$ (7 pieces) and this is the lower bound. In the worst case, the set of pieces possessed by any peer is a subset of the set of pieces possessed by the peer with the maximum progress. In this case, we need to download all missing pieces of peer 1 and it is equal to $d = 1 - \max_{i=1}^{3}\{v_i\} = 0.7$ (14 pieces), which is the upper bound. The remaining question is how to uncondition the number of peers and v_i's. We are now in the position to develop the mathematical model. As mentioned before, we consider two scenarios: regular peer arrival case and flash crowd case. Let us first focus on the analysis of regular arrival case.

Figure 3.1: Illustration of the lower and the upper bound of cross-ISP traffic

3.1.2 Homogeneous Case Analysis

Let us first consider the homogeneous case: the file downloading time is the same for all peers. Without loss of generality, assume the file size is 1 and the file downloading time is T. We have the following result.

Theorem 7 *For a given ISP in which all peers use an ISP-friendly file distribution protocol, if there is no seeder in that ISP, peers arrival process is characterized by a Poisson process with an average rate λ and all peers in that ISP have the same downloading time T, then the average amount of incoming cross-ISP traffic caused by each peer in the steady state, denoted by $E(d)$, is lower bounded by*

$$E(d) \geq e^{-\bar{n}} \bar{n}^{-1/2} I_1(2\sqrt{\bar{n}}),$$

where $\bar{n} = \lambda T$ and $I_1(x)$ is the modified Bessel function.

Note that $\bar{n} = \lambda T$ is the average number of peers in that ISP, and this lower bound is a *decreasing* function of $\bar{n}$.

Proof: Please refer to Appendix for derivation. ∎

Assume that each peer is aware of other peers' state in real time, then for a P2P system which follows the ELP, one can derive an upper bound of the average cross-ISP traffic as follows.

Theorem 8 *For a given ISP in which all peers use an ISP-friendly file distribution protocol, if there is no seeder in that ISP, peers arrival process is characterized by a Poisson process with an average rate λ and all peers in that ISP have the same downloading time T, then the average amount of incoming cross-ISP traffic caused by each peer in the steady state, denoted by $E(d)$, is upper bounded by*

$$E(d) \leq \frac{1}{\bar{n}}(1 - e^{-\bar{n}}),$$

where $\bar{n} = \lambda T$.

Proof: Please refer to Appendix for derivation. ∎

3.1.3 Heterogeneous Case Analysis

In here, we extend our model to consider the heterogeneous case where peers have different downloading time.

In a large P2P system, the total service capacity of the system scales up as the number of peers increases [34], and the downloading time is roughly independent of the number of peers in the system. We use T, which is now a random variable, to represent the file downloading time of a peer and extend the model to derive the bounds of the heterogeneous case.

Theorem 9 *For a given ISP in which all peers use an ISP-friendly file distribution protocol, there is no seeder in that ISP and the peers arrival process is characterized by a Poisson process with an average rate λ. Let q_i be the probability that the downloading time for a peer will be τ_i, then the average amount of incoming cross-ISP traffic caused by each peer in the steady state, denoted by $E(d)$, is bounded by*

$$e^{-\bar{n}}\bar{n}^{-1/2}I_1(2\sqrt{\bar{n}}) \leq E(d) \leq \frac{1}{\bar{n}}(1 - e^{-\bar{n}}),$$

where $\bar{n} = \lambda q_1\tau_1 + \lambda q_2\tau_2 + \ldots$, and $I_1(x)$ is the modified Bessel function.

Proof: Please refer to Appendix for derivation. ∎

Remark: In summary, Theorem 9 gives the lower bound and upper bound of the average cross-ISP traffic caused by each peer when all peers adopt the ELP. To illustrate the spread of these bounds, we consider an ISP with different values of $\bar{n}$. Figure 3.2 illustrates the spread of these two bounds on the cross-ISP traffic, as well as the average cross traffic when one uses the random downloading strategy (e.g., the conventional

P2P file distribution protocol) with $N = 200$ peers in the P2P system. One can observe that both bounds decrease quickly when $\bar{n}$, the average number of peers within that ISP, increases. Notice that the cross-ISP traffic for random downloading remains high. This justifies the design and implementation of an ISP-friendly file distribution protocol.

Someone may notice that the upper bound is almost equal to $1/\bar{n}$ and argue why not just design a gateway-architecture to achieve this bound: at any moment for a given ISP, there is only one peer which connects to external peers and all other internal peers subscribe to this peer. So why we still need to design a new ISP-friendly protocol? Because the gateway-architecture has several disadvantages: a) It needs someone to organize the architecture. b) It is difficult to deal with selfish behaviors, e.g. the gateway peer may not want to upload to other internal peers. c) More important, it is not compatible with current BitTorrent protocol so that it could not be deployed gradually. We aim to design an ISP-friendly protocol to overcome these problems.

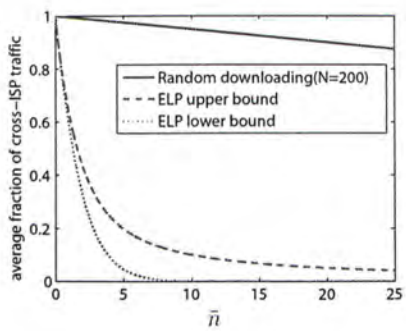


Figure 3.2: Average fraction of cross-ISP traffic vs. the average number of peers in the ISP

3.1.4 Flash Crowd Analysis

Let us now consider the flash crowd scenario when a large number of peers arrive to the ISP in a very short period of time. This occurs, for example, when a very popular movie or an OS kernel update is being first published to the Internet. Based on the same assumptions we made in the regular arrival analysis (except that peer arrival process is no longer Poisson), we can derive the upper and lower bound of the cross-ISP traffic.

Theorem 10 *For a given ISP in which all peers use an ISP-friendly file distribution protocol and there is no seeder in that ISP. At time $t = 0$, n peers arrive and there is no more peer arrival after $t > 0$. Let τ_{min} (τ_{max}) be the shortest (longest) downloading time of these peers. The average*

amount of incoming cross-ISP traffic caused by each peer, denoted by $E(d)$*, is bounded by*

$$1/n \leq E(d) \leq \left(1 + \log\left(\frac{\tau_{max}}{\tau_{min}}\right)\right)/n.$$

Proof: Please refer to Appendix for derivation. ∎

Remark: Notice that τ_{max} is the downloading time of peers with the lowest downloading rate, τ_{min} is the downloading time of peers with the highest downloading rate. It is interesting to observe that the upper bound of the cross-ISP traffic depends on τ_{max}/τ_{min} and n only.

3.2 An ISP-friendly BitTorrent Protocol

In this section, we present our *ISP-friendly file distribution protocol* which uses the ELP to reduce the cross-ISP traffic. To appreciate the proposed protocol, we first provide a brief review of the BitTorrent (BT) protocol. Note that one design requirement of our protocol is that it has to be "*compatible*" with the current BitTorrent software and our clients can communicate directly with existing BT peers. This feature is particularly important since this new service can then be incrementally deployed.

Under the BT protocol, a file is to be divided into many non-overlapping pieces (the default size is 256 KB) and there is at least one peer, which is called a seeder, who holds all these pieces and this seeder wants to publish the file. A peer can get the file either from the seeder, or from other peers holding those pieces it does not possess. Each peer offers upload service to other peers only to the extend that the service is reciprocated. By coupling the service each peer can receive to its upload contribution, the BT protocol successfully makes each peer play a role of a server and thereby improve the performance of the system. There is a special node called the *tracker*, which keeps track of all peers in the system. A peer needs to first contact the tracker to get a subset of peers who are downloading the file. This peer then establishes connections to other peers and finds out what pieces these peers possess. Then this peer will send out an INTERESTED message to its connected peers, indicating that there exists some pieces it does not possess and this peer wishes to receive some download service. One important point is that the INTERESTED message does *not* indicate which piece this peer wants. The piece selection is determined in later step.

Uploading is called *unchoking* in BitTorrent. Each peer unchokes a fixed number of peers simultaneously (the default number is four). Which peers to unchoke is determined by the current downloading rate from these peers, i.e., each peer uploads to the four peers who provide it with the top four downloading rates. This unchoking mechanism is called the

tit-for-tat policy, and one implication of this policy is that it deters free-riding. Beside the tit-for-tat policy, there is another unchoking mechanism called the *optimistic unchoking*, which allows each peer to explore the downloading rates of other peers. Under the optimistic unchoking, each peer randomly selects another peer to upload without considering the service contribution of the selected peer. Optimistic unchoking serves two purposes: (1) it helps new peers to get some pieces so that they can contribute to the community, and (2) it is an attempt to discover another peer with a higher uploading rate. If this kind of peer is found, then the peer with the smallest downloading rate in the regular unchoking set will be replaced by this peer.

Downloading in BitTorrent is determined by the piece selection policy called the *local rarest first*. When a peer is ready to download from another peer, usually there are several potential choices of pieces to download. Under the local rarest first strategy, a peer will choose the piece which has the least number of copies among its connected neighbors to download first. The local rarest first policy not only can balance the distribution of pieces in the system, but can also enhance the overall file availability.

Let us now present our ISP-friendly protocol. In essence, it is a variant of the BitTorrent protocol which exploits ELP. The goal is to reduce the amount of cross-ISP traffic and at the same time, maintain good performance (e.g., small file downloading time). There are many details in our protocol, but the basic idea is: *a peer will not download a piece from external neighbors if he finds that this piece is held by some internal neighbors.*

To adopt ELP, it is necessary for a peer to distinguish peers that are within the ISP and peers that reside in other ISPs. For a BitTorrent peer, it obtains the IP addresses of its connected neighbors from the tracker. Therefore, a peer needs to find the relationship between an IP address and its associated ISP. This type of association can be easily constructed using tools like the ASFinder in the CoralReef suite [3]. In fact, an ISP can set up a "whois" server to provide this mapping service to all peers within its domain. It only needs to map all IP addresses belonging to itself and its customer ISPs as internal peers, and this can be easily constructed using the CIDR address format. An important point is that there is an *economic incentive* for an ISP to provide this type of mapping service. It can encourage peers to use the ISP-friendly protocol, therefore reduce the cross-ISPs traffic and its operating cost.

Being able to distinguish between internal peers and external peers, each peer can exploit the ELP via the following steps:

1. Divides its neighbors into two type, *internal neighbors* are the neighboring peers which belong to the same ISP as itself, and *external neighbors* are the neighboring peers which belong to other ISPs.

2. Creates a list C_I where $C_I[j]$ records the number of copies of the j^{th} piece that are within the *internal neighbors*. Similarly, creates a list C_E where $C_E[j]$ records the number of copies of the j^{th} piece that are within the *external neighbors*.

3. For a given peer, let $\mathcal{F}_L$ denote the set of pieces held by this peer (or localhost). For a neighboring peer, let $\mathcal{F}_R$ denote the set of pieces held by this neighbor. If it is an internal neighbor, sends an INTERESTED message to it if it has some pieces which are not possessed by the localhost, i.e., $\mathcal{F}_R \backslash \mathcal{F}_L \neq \emptyset$. If it is an external neighbor, sends an INTERESTED message to it if it has some pieces which are not possessed by *all* internal peers, i.e., $C_I[j] = 0$ for some $j \in \mathcal{F}_R \backslash \mathcal{F}_L$.

4. Upon an unchoking event, the peer has to handle it differently depending on whether it was unchoked by an internal neighbor or external neighbor. If the peer was unchoked by an internal neighbor, the peer will request a piece k using the local rarest first policy over C_I:

$$k = \arg\min_j \{C_I[j]\}, \quad j \in \mathcal{F}_R \backslash \mathcal{F}_L. \tag{3.2}$$

   If the peer was unchoked by an external neighbor, the peer will request only those pieces which are not available in the internal neighbors and using the local rarest first policy over C_E:

$$k = \arg\min_j \{C_E[j]\}, \quad j \in \mathcal{F}_R \backslash \mathcal{F}_L, C_I[j] = 0. \tag{3.3}$$

All other parts of the ISP-friendly protocol remain the same as the current BitTorrent protocol, e.g., tracking, tit-for-tat, optimistic unchoking and so forth.

According to the above mentioned modifications, whether piece k is a potential choice for downloading from a neighboring peer can be determined by the following decision function:

```
def want(k):
    return k ∈ F_R \ F_L and
        (ISP_neighbor == ISP_localhost or C_I[k] == 0)
```

If want(k) returns "False" for all piece index k, then the peer is *not interested* in this neighbor. If it returns "True" for some piece index k, then the peer will send an INTERESTED message to this neighbor and wait to be unchoked.

Upon unchoked by an internal (external) neighbor, the peer can use the function want(k) to find out all potential pieces to request, and then look up the table C_I (C_E) to determine which piece to request first based

on the local rarest first policy. Notice that when all neighbors are internal neighbors or all neighbors are external neighbors, this ISP-friendly protocol behaves exactly the same as the current BitTorrent protocol.

In summary, the ISP-friendly protocol proposed above uses the ELP to send the INTEREST message, and during the piece selection process, uses the ELP and the local rarest first policy. By doing so, a peer determine which peers to download from and also attempts to avoid downloading any duplicate piece which resides within the same ISP.

Before we leave this section, it is important for us to comment about the difference between the proposed ISP-friendly protocol and the idealized model as presented in Sec. 3.1. In practice, the BitTorrent protocol (and the proposed ISP-friendly protocol) is quite involved. It contains many mechanisms to ensure good performance, such as *random first piece selection*, *endgame mode*, *anti-snubbing* and so on. Furthermore, each peer only has a partial view of the whole P2P system and can only make decisions based on its local information. In addition, it takes time for information (e.g., piece availability) to be propagated throughout the P2P network. Therefore, this ISP-friendly protocol may deviate from the ELP in the sense that

- Each peer may not be connected to all internal peers.
- The piece availability information cannot be updated instantaneously.

The above scenarios may lead to the situation that duplicated pieces could be downloaded from external peers. The impact of the first scenario can be reduced if peers can contact the tracker more often to request for more neighbors. The impact of the second scenario can be reduced if peers can update their local information (e.g., piece availability) more frequently with each other.

Notice that the ISP-friendly protocol only aims at reducing the *incoming* cross-ISP traffic. By doing so, it also reduces the *outgoing* cross-ISP traffic because of the built-in tit-for-tat mechanism in BitTorrent. This mechanism enforces certain degree of fairness in data exchange and therefore the total amount of outgoing cross-ISP traffic is approximately equal to the incoming cross-ISP traffic. This is verified by our experiments which are presented in the following section.

3.3 Performance Evaluation & Measurements

In order to evaluate the cross-ISP traffic reduction and the average file downloading time of the proposed ISP-friendly protocol, we modify a BitTorrent software to implement the ISP-friendly features mentioned in Sec. 3.2 and carry out experiments and measurements on the PlanetLab. To compare the proposed ISP-friendly protocol to the current BitTorrent

protocol, we also instrument the same BitTorrent software to collect traffic information for comparison. In the following, we describe in detail on how we carry out the experiment.

3.3.1 Choice of the BitTorrent Client

The first BitTorrent client was developed by Bram Cohen, the inventor of the BitTorrent protocol [9]. Note that there are many other BitTorrent clients available, such as μTorrent, BitComet, Azureus and so on. Since there is no de facto standard, Cohen's BitTorrent client is considered as the reference for the BitTorrent protocol. Thus, this client is also called the "Official BitTorrent client". It is an open source software, written in Python and can be executed on many different platforms. Most BitTorrent clients maintain compatibility with the official BitTorrent client. The main differences of these clients are the user interface, configuration options (e.g., caching option to reduce disk access) and certain extensions to the BitTorrent protocol (e.g., UDP transport to traverse NAT). Our goal is to evaluate the basic BitTorrent protocol and the proposed ISP-friendly BitTorrent protocol. Thus, we choose the official BitTorrent client and we instrument the official BitTorrent client version 4.4.0 which was released in 2006.

3.3.2 Experimental Setup

We carry out experiments under two scenarios: regular peer arrival and flash crowd. For each scenario, we run the experiment twice with the same settings, one with the official BitTorrent client, the other one with the ISP-friendly BitTorrent client, thus there will be four experiments in total. In order to compare their cross-ISP traffic and the file downloading performance, each client logs at least the following information: starting time, ending time, bytes downloaded from internal/external neighbors, bytes uploaded to internal/external neighbors.

There are many configuration options for the official BitTorrent clients. The main default parameters are: the maximum upload rate (default is 20 KB/s), the maximum number of peers to upload to (default is 4), the number of pieces downloaded before switching from random to rarest first piece selection (default is 4), time interval to request more peers from the tracker (default is 300 secs.), the minimum number of neighbors before requesting more peers from the tracker (default is 20), the maximum number of neighbors (default is 80) and so on. It is outside the scope of this study to evaluate the impact of each BitTorrent's parameter. In our experiments, we use the default parameters except that: the time interval to request more peers from the tracker is set to 60 seconds, the minimum number of neighbors before requesting more peers from the tracker is set to 80. We set these two parameters to help peers discover

other peers and connect to them sooner.

The typical file size of a BT file distribution ranges from tens to hundreds megabytes (files can be music albums, TV shows, movies and so on). Usually users will set the maximum uploading rate larger than the default setting 20KB/s to speed up their downloading. To avoid consuming too much bandwidth and other resource of the PlanetLab nodes, we use a relatively small file (20MB) for downloading, and the piece size is also scaled down to 32KB. There is a seeder in the system to ensure file availability in all our experiments. To avoid the seeder become the bottleneck, its maximum uploading rate is set to 50KB/s, larger than the maximum uploading rate of other peers.

Since most nodes in the PlanetLab are within universities, one can consider each university as an "ISP", and construct a database to map each PlanetLab node to "ISP"(There are some differences between "AS" and "ISP", but it does not matter to our experiments, or we may call it "AS-friendly protocol"). In our experiments, we consider six "ISPs": Berkeley (16 nodes), Columbia (3 nodes), Cornell (6 nodes), MIT (7 nodes), Princeton (11 nodes), and OTHER (32 nodes). Since there may be more than 60 peers for some experiments, we may assign several peers to the same node. But to avoid overloading the node, no more than three peers will be running on the same node at any time.

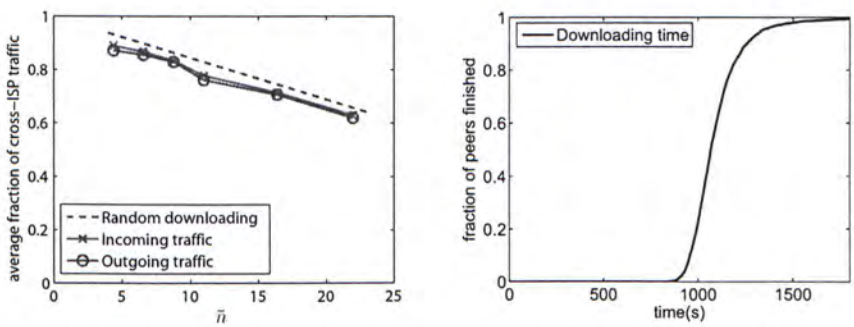
3.3.3 Regular Peer Arrival

In the following experiments, we study the cross-ISP traffic and the file downloading time of the official BitTorrent and the proposed ISP-friendly BitTorrent in regular peer arrival scenario, i.e., peer arrival to the ISP is a Poisson process. To carry out meaningful and realistic experiments, we instrument each ISP with a different peer arrival rate and peers from different ISPs participate in the same torrent file sharing. Note that we have six ISPs: Berkeley, Columbia, Cornell, MIT, Princeton, OTHER. In our experiments, we initiate the seeder and the tracker in Columbia and there is no other peer in Columbia. Peers are launched in the other five ISPs according to Poisson processes. We know that the sum of several independent Poisson arrival streams is still Poisson arrival, thus the peer arrival for the whole P2P network(containing five ISPs) is still Poisson. We carry out the experiment multiple times with the peer's average interarrival time as $250s$, $167s$, $125s$, $100s$, $67s$ and $50s$ respectively for a certain ISP(we choose Berkeley), and the peer arrival for other ISPs are adjusted accordingly to make sure that the peer arrival for the whole P2P network is a Poisson process with an average interarrival time being $16s$. This implies that the ratio of peers in Berkeley and the peers in the whole P2P networks will be about 4/64, 6/64, 8/64, 10/64, 15/64 and 20/64 respectively. The experiment lasts for 48 hours each time. With the log file, we can calculate the average downloading time T in Berkeley,

and then derive the average number of peers by $\bar{n} = \lambda T$.

Experiment 1: Regular Peer Arrival for Official BitTorrent

We carry out the experiment using the official BitTorrent client with the settings mentioned above. Since the *maximum* uploading rate of a peer is 20KB/s, and there is only one seeder in the system whose upload rate is negligible comparing to the aggregate upload rates of all peers, therefore, the expected downloading rate of a peer in the system is upper bounded by 20 KB/s. For the experiment, the size of the published file is 20MB, thus the average file downloading time would be larger than $1000s$. This is confirmed by our experiment. Figure 3.3 illustrates the experimental results.



(a) Cross-ISP traffic vs. # of peers

(b) CDF of downloading time

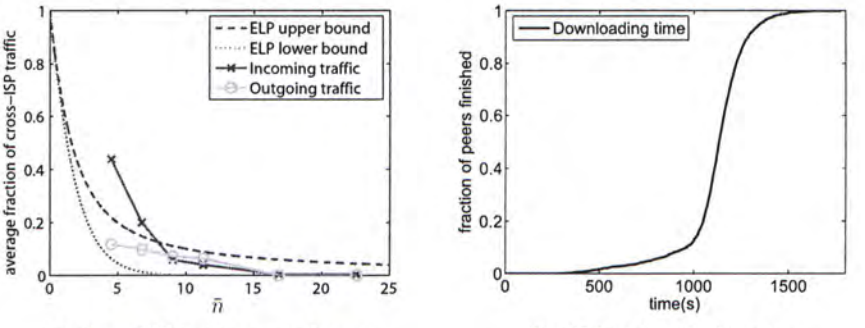
Figure 3.3: Performance of the Official BitTorrent under Steady Peer Arrival

Figure 3.3(a) shows the average fraction of incoming and outgoing cross-ISP traffic generated by each peer in Berkeley with different average interarrival time. In Equation (3.1), we show the fraction of cross-ISPs traffic for the *random downloading* strategy and we also plot this curve in the figure. As stated in Equation (3.1), the expression is $f = 1 - n/N$ where n is the average number of peers in a certain ISP(It is Berkeley here.) and N is the average number of peers in the whole P2P system. Both n and N can be calculated by the average interarrival time and the average downloading time. From the figure, one can observe that the cross-ISP traffic generated by the official BitTorrent client is *very similar* to the random downloading strategy. It generates a lot of incoming and outgoing cross-ISP traffic. One can also observe that outgoing traffic is slightly less than the incoming traffic. The reason is that there is a seeder in the system and this seeder uploads to other peers but never perform any downloading. Therefore, other ISPs observe more incoming cross-ISP traffic.

Figure 3.3(b) shows the cumulative distribution function (CDF) of the file downloading time for Berkeley. It can be seen that the curve is sharp, which means that the downloading time for most peers are roughly the same.

Experiment 2: Regular Peer Arrival for the ISP-friendly Protocol

We use the same setting as Experiment 1 except the clients are replaced by our ISP-friendly clients discussed in Section 3.2. The results are illustrated in Figure 3.4.



(a) Cross-ISP traffic vs. # of peers

(b) CDF of downloading time

Figure 3.4: Performance of the ISP-friendly BitTorrent under Steady Peer Arrival

Figure 3.4(a) shows the average fraction of incoming and outgoing cross-ISP traffic generated by each peer in Berkeley with different average interarrival time using ISP-friendly protocol. We also show the lower and upper bounds of the derived cross-ISP traffic model. One can observe that the cross-ISP traffic is *greatly reduced* compared to the official BitTorrent client. The experiment curve for the incoming traffic falls between the bounds when $\bar{n}$, the average number of peers in Berkeley is larger than 7. When $\bar{n}$ is small, the experiment curve exceeds the upper bound. The reason is that the peers in Berkeley are so rare compared to the whole P2P system, it is usually difficult for a newly arriving peer in Berkeley to discover and establish connection to other peers within Berkeley soon. Then this newly arriving peer may request pieces from external peers even these pieces are held by some internal peers, resulting an increase in the cross-ISP traffic. However, if $\bar{n}$ is small in Berkeley, the aggregate cross-ISP traffic will not be very significant. Notice that the ISP-friendly protocol differs from the official BitTorrent client only in the downloading strategy. However, the outgoing cross-ISP traffic is also *significantly reduced*. It is interesting to observe that the outgoing traffic is much less than the incoming traffic when $\bar{n}$ is small, and it can be interpreted like this: the newly arriving peer in Berkeley performs little uploading to external peers compared to downloading, since it has not many pieces to upload.

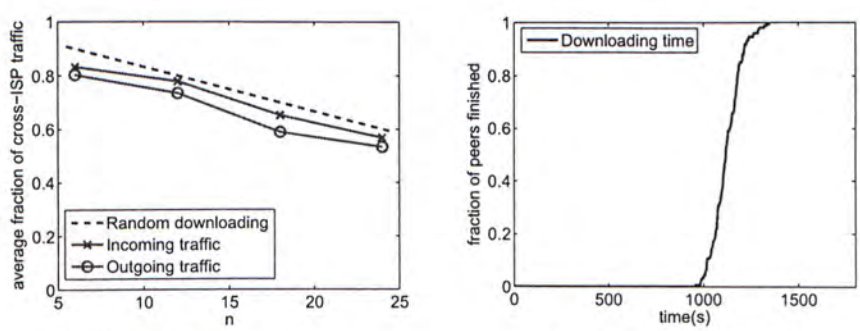
Figure 3.4(b) shows the cumulative distribution function (CDF) of the file downloading time for Berkeley. The first observation is that the downloading time is slightly larger($< 10\%$) than the official BitTorrent. There are two reasons for the increase in file downloading time. First,

since peers follow the ELP, the seeder, which resides in a different ISP, may remain idle since downloading from seeder is considered as cross-ISP traffic. Second, since some pieces can only be downloaded from internal peers according to the ELP, it will also degrade some downloading chance. However, the gap is not very large and it will be reduced if there are more peers within Berkeley. Another observation is that the variance of the file downloading time is a little larger than the official BitTorrent.

3.3.4 Flash Crowd

In here, we study the cross-ISP traffic and the file downloading time of the official BitTorrent and the ISP-friendly BitTorrent under the flash crowd scenario. There are five ISPs: a seeder and a tracker are located in Columbia. All peers arrive at $t = 0$ to the four ISPs. Number of peers in the ISPs are: 6 (Cornell), 12 (MIT), 18 (Princeton) and 24 (Berkeley). We run each experiment multiple times to obtain a good confidence interval.

Experiment 3: Flash Crowd for Official BitTorrent
We use the official BitTorrent clients in this experiment. The results are shown in Figure 3.5.



(a) Cross-ISP traffic vs. # of peers

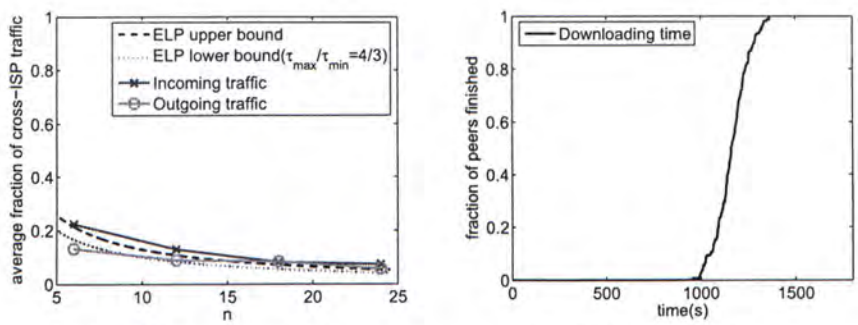
(b) CDF of downloading time

Figure 3.5: Performance of the Official BitTorrent under Flash Crowd

Figure 3.5(a) shows the average cross-ISP traffic generated by each peer in different ISPs. The total number of peers in the system is 60, thus we plot the curve $f = 1 - n/60$ (using Eq. (3.1)) to represent the random downloading strategy. One can observe that the cross-ISP traffic generated by the official BitTorrent client is very close to Eq. (3.1). It means that the official BitTorrent client also generates significant amount of cross-ISP traffic in the flash crowd scenario. Another observation is that the outgoing cross-ISP traffic is slightly less than the incoming cross-ISP traffic. This is justified due to the tit-for-tat policy in BitTorrent. The reason for the slight difference is that there is a seeder in the system who uploads to other peers but never perform any downloading.

Figure 3.5(b) shows the CDF of the file downloading time. The results indicate that it is very "deterministic" in the sense that most peers finish the file download approximately at the same time.

Experiment 4: Flash Crowd for the ISP-friendly Protocol
The setting of this experiment is exactly the same as Experiment 3 except we use the ISP-friendly client. The results are shown in Figure 3.6.



(a) Cross-ISP traffic vs. # of peers

(b) CDF of downloading time

Figure 3.6: Performance of the ISP-friendly BitTorrent under Flash Crowd

Figure 3.6(a) shows the average cross-ISP traffic generated by each peer in different ISPs. One can observe that the cross-ISP traffic is *significantly reduced* compared to the official BitTorrent client (Figure 3.5(a)). We can also observe that outgoing cross-ISP traffic is slightly less than incoming cross-ISP traffic due to the tit-for-tat policy.

Figure 3.6(b) shows the cumulative distribution function of the file downloading time. Again, it is very deterministic in that most peers can finish the file download around the same time. Compared with Figure 3.5(b), one can observe that the file downloading time of the ISP-friendly client is only *slightly* worse ($< 5\%$) than the official BT client.

3.4 Black Hole Security Attack

We have seen that the ISP-friendly protocol can greatly reduce the cross-ISP traffic while keeping good file downloading performance. In this section, we present the "*black hole attack*", which may have a detrimental effect on the ISP-friendly file distribution protocol.

Consider a free-rider in a file distribution session. This free-rider will advertise to other peers that it has a lot of pieces (or all pieces of the file) but it refuses to provide any upload service to other peers. This type of free-riders do exist in the current BitTorrent file distribution but they only receive minimal amount of service: free riders can only download pieces via the the "optimistic unchoked" connection. Therefore, the file downloading time of these free-riders is significantly larger than those

normal peers who are willing to provide upload service. This type of free-riding, however, can be *detrimental* to the ISP-friendly protocol. In particular, when the free-rider announces that it has all pieces of the file (or it pretends to be a seeder), it prevents other internal peers obtaining information from external peers, and this may halt the whole file download process within the ISP. We call this the "*black hole attack*".

To overcome the black hole attack, one needs to provide some mechanism to filter out the attackers or peers with very low uploading rate. One may first consider the black listing technique to do the peers filtering. But black listing needs addition collaboration among peers since one peer could not detect the attacker based on his own local view. This addition collaboration, will make the ISP-friendly protocol incompatible with the current BitTorrent protocol. Instead, we propose an *Enhanced ISP-friendly protocol* which can filter bad peers effectively while keeping the compatibility with the current BitTorrent protocol.

Similar to the ISP-friendly protocol proposed in Section 3.2, each peer classifies its neighbors into two categories: *internal peers* and *external peers*. In the *Enhanced ISP-friendly protocol*, each peer will pick less than or equal to q internal peers as its *active co-agents* (we will show how to select *active co-agents* later). Denote the following set:

$$S = \{c | \text{piece c is missed by all its } \textit{active co-agents}\}.$$

the only thing we need to modify compared to the previous ISP-friendly protocol is the decision function $want(k)$. Whether piece k is a potential choice for downloading from a neighbor can be determined by the following new decision function:

```
def want(k):
    return k ∈ F_R \ F_L and
        (ISP_neighbor == ISP_localhost or k ∈ S)
```

If want(k) returns "False" for all piece index k, then the peer is *not interested* in this neighbor. If it returns "True" for some piece index k, then the peer will send an INTERESTED to this neighbor and wait to be unchoked. Upon unchoked by an internal (external) neighbor, the peer can use the new function want(k) to find out all potential pieces to request, and then look up the table C_I (C_E) to determine which piece to request first based on the local rarest first policy.

Now let us discuss how to pick the *active co-agents*. The intuitive notion of *active coagents* is that, if peer A considers peer B as its *active co-agent*, it implies that peer A detects that peer B works well on uploading to internal peers, thus it may not be necessary for peer A to download the pieces which are held by B from external peers. As we emphasized earlier, we want to design a protocol which is compatible with the current BitTorrent protocol, thus we can pick the *active co-agent* based on the

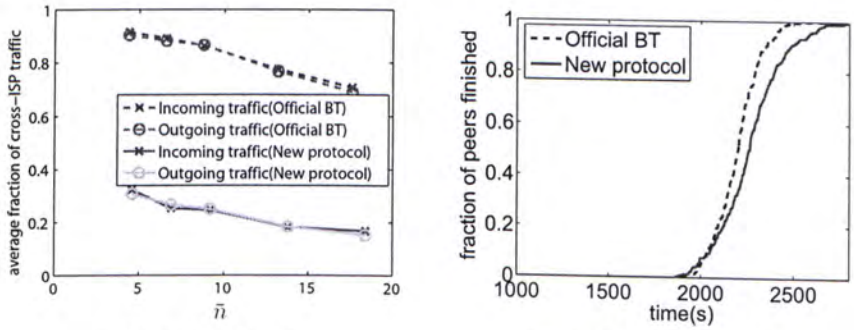
local information only. A credible evidence that an internal peer works well on uploading to internal peers is that it uploads to you recently. Based on this notion, we develop the selection algorithm as follows.

1. Measuring the downloading rate r_i from each internal peer, where r_i is the average rate for the last T seconds.
2. Ranking the internal peers in a list according to r_i in decreasing order.
3. Truncating the list for peers with $r_i < R$.
4. Picking the top q peers as *active co-agents*.

There are three parameters in this algorithm, T, R and q. We do not want the *active co-agents* to change too rapidly. So we select T to be within 1 to 2 minutes. Threshold R is to prevent anti-snubbing attack in which a peer schedules to satisfy just one request per 60 seconds to avoid getting snubbed. A reasonable value for R is between $0.5KB/s - 2KB/s$. Lastly, q is crucial to reducing the cross-ISP traffic. Our measurement study shows that it is sufficient to set $q \geq 5$. To evaluate the enhanced ISP-friendly protocol, we carry out the regular peer arrival experiments as follows.

We configure our enhanced ISP-friendly client to the same settings as Section3.3(e.g., maximum uploading rate is $20KB/s$, maximum number of peers to upload to is 4, etc). But the size of the file for downloading is now doubled to $40MB$. The peer arrival for the whole P2P network is a Poisson process with average interarrival time being $31s$. We carry out the experiment multiple times with the peer's average interarrival time as $500s$, $333s$, $250s$, $166s$ and $125s$ respectively for Berkeley. This implies that the ratio of peers in Berkeley and the peers in the whole P2P network is about 2/32, 3/32, 4/32, 6/32 and 8/32 respectively. In the experiment, there is always one malicious free rider (or faked seeder) in Berkeley. The three parameters of the enhanced ISP-friendly protocol are set as $T = 2min$, $R = 0.5KB/s$ and $q = 10$. The results are shown in Figure 3.7.

In Figure 3.7(a), one can observe that the cross-ISP traffic is significantly reduced compared to the official BitTorrent(reduced to about 1/3 in our experiments). In Figure 3.7(b), one can see that there is only a slight performance degradation ($< 10\%$) in the file downloading time compared to the official BitTorrent. Notice that the performance of the enhanced ISP-friendly protocol is not seriously affected by the malicious free rider. The slight performance degradation in file downloading is acceptable, given that the large reduction of cross-ISP traffic. And the performance gap of file downloading time can be even reduced if there are more peers in Berkeley. Actually the experiment results are similar



(a) cross-ISP traffic

(b) CDF of downloading time

Figure 3.7: Enhanced ISP-friendly BitTorrent under Regular Arrival

to Figure 3.7 when there are several malicious peers. And the enhanced ISP-friendly protocol works well in the Flash Crowd experiments too.

□ **End of chapter.**

Chapter 4

Related Work

There are numerous empirical studies on the BT protocol, for instance, [4, 10, 18, 23, 29]. Izal et al. [18] present the traffic information on peers behavior collected during a five-month period. Pouwelse et al. [29] study the availability, the integrity, the flash crowd effect and the download performance from a trace which was collected for eight months. Erman et al. [10] study the session interarrival times, sizes and durations and propose to use the hyper-exponential distribution to model the session interarrivals, and use the log-normal distribution to fit session durations and sizes. Legout et al. [23] evaluate the two core components of BitTorrent: choking and the rarest first algorithm and claim that they are enough to guarantee the efficiency and viability. Bindal et al. [4] report great variability of downloading time and claim that instead of network bandwidth, *"close neighbor set"*(i.e. those peers in a stable data-exchange relationship) is the major contributing factor for the variability. However, a major limitation of these empirical studies is that the data collected is usually from a local view (i.e. the tracker log or a modified client), and the behavior is very time-dependent. Therefore, it is not an easy task to understand the efficiency of the BT protocol simply based on empirical studies.

There are also several analytical studies of BT protocol. Yang et al. [34] study the service capacity of BT protocols. Their result indicates the service capacity of BT protocols increases exponentially at the beginning and scales well with the number of peers, thus providing fast downloading independent of demand rate. Qiu et al. [30] extend the coarse-grain Markovian model in [34] by providing an analytical solution to a fluid model in steady state which shows high scalability and stability of BT protocols. Our work differs from [30, 34] in that we provide a detailed probabilistic model to capture the peers' diversity (in terms of downloading progress) and show the change of downloading speed during the whole session. We also analyze the peer selection and chunk selection which are not considered in [30, 34]. Fan et al. [13] also generalize Qiu's model by dividing peers into three types according to number of

chunks they hold. Our work extends the number of types from 3 to $K-1$ so as to capture the system more accurately. Under the assumption that "uplink is the only constraint", Mundinger et al. [28] propose a deterministic scheduling algorithm to achieve the optimal makespan which requires global knowledge. Sanghavi et al. [33] also propose a gossip-like randomized algorithm requiring only local knowledge. Both studies in [28] and [33] are orthogonal to ours as they only consider the "closed system" where no new peer will arrive during the file dissemination while we consider an "open system" which new peers are joining in according to Poisson process. The work that is closely related to our study is [27]. In that paper, the authors provide a detailed probabilistic model to investigate the stability and effectiveness of a peer-to-peer file swarming system. Their results state that even by the "random chunk selection" policy, the system throughout is still asymptotically optimal. Our work improves and extends the result in [27] by providing tighter asymptotic bounds and relaxing its assumption of unlimited upload capacity. Moreover, we study the peer selection by both random selection and coordinated matching policies. Gaeta et al. [14] also use a probabilistic model to study the large-scale P2P network but they are focusing on searching strategy. There are some other analytical studies in fairness of BT besides performance modeling. In [7,25,26], the authors present a mathematical analysis on service differentiation in resource allocation for P2P networks. In [11], the authors present a mathematical framework to study the tradeoff between performance and fairness in BT-like systems. In [35], authors present the first analytical model of BT-like systems and quantify the tradeoff between scalability and QoS support for multimedia streaming applications.

There are only few studies addressing the issue of cross-ISP traffic. One approach is caching [6] but one has to address the copyright legal issue. In [19], authors propose to place some "gateway peers" to connect to external peers and other peers only download within the ISP. However, one has to address the issue of service availability due to sudden departure of gateway peers. Authors in [5] examine a technique named "biased neighbor selection" to explore traffic reduction, but the study was only carried out via simulation. In our work, we propose to exploit the *content locality* which requires no extra hardware investment from the ISP. We analytically evaluate the cross-ISP traffic reduction, and at the same time, propose and implement such mechanism to achieve the reduction while keeping good downloading performance.

□ **End of chapter.**

Chapter 5

Conclusion

In this thesis, we propose a probabilistic model which generalizes the model in [27] to capture the basic properties of a file swarming system. Under the same assumption as [27](i.e. unlimited upload capacity), we first improve its asymptotic bound of the average downloading time. Then we provide two different approaches, namely fetching multiple bitmaps and using FEC code, to help the system achieve nearly optimal performance. Besides showing that FEC code can also remedy the last-piece problem, we also remove the assumption of "unlimited upload capacity" and analyze the performance under the random peer selection algorithm. Since the performance deteriorates due to request collision, we propose a matching scheme to improve the performance. We show that under the coordinated matching, if peers are altruistic the system performance can achieve as good as the system with unlimited upload capacity. Even when the system deploys certain incentive mechanism (tit-for-tat), the average downloading time is still good. The result suggests that the performance of a peer-to-peer file swarming system does not depend critically on altruistic peers, but rather due to the diversity of peers stored data so the system can achieve good performance. We also address how one can reduce the cross-ISP traffic for file distribution applications. We use a simple and effective idea: exploit the content locality to reduce the traffic. We analytical show the significant cross-ISP traffic reduction when one uses the above principle, and then design and implement such mechanism on a BT software, carry out extensive experiments and measurements on the PlanetLab to demonstrate its effectiveness. Lastly, we illustrate the black hole security attack and how one can modify the proposed protocol to address this problem.

□ **End of chapter.**

Bibliography

[1] http://multiprobe.ewi.tudelft.nl/dataset.html.

[2] Bittorrent protocol. http://www.bittorrent.com/protocol.html.

[3] CAIDA. CoralReef suite. http://www.caida.org/tools/measurement/coralreef.

[4] R. Bindal and P. Cao. Can self-organizing P2P file distribution provide qos guarantees? In *OSR Special Issue on Self-Organizing Systems*, 2006.

[5] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in bittorrent via biased neighbor selection. In *IEEE ICDCS*, 2006.

[6] CacheLogic. Advanced solutions for peer-to-peer networks. http://www.cachelogic.com/.

[7] F. Clévenot-Perronnin and K. R. P. Nain. Multiclass P2P networks: Static resource allocation for service differentiation and bandwidth diversity. In *IFIP WG7.3 Performance*, Juan-les-Pins, France, 2005.

[8] B. Cohen. Incentives build robustness in bittorrent. http://www.bittorrent.org/bittorrentecon.pdf, May 2003.

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

[10] D. Erman, D. Ilie, and A. Popescu. Bittorrent session characteristics and models. In *HET-NETs '05, Third International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks*, Ilkley, United Kingdom, 2005.

[11] B. Fan, D.-M. Chiu, and J. C. S. Lui. The delicate tradeoffs in bittorrent-like file sharing protocol design. In *ICNP, 2006.*

[12] B. Fan, D. M. Chiu, and J. C. S. Lui. The delicate tradeoffs in bittorrent-like file sharing protocol design. In *IEEE ICNP*, 2006.

[13] B. Fan, D.-M. Chiu, and J. C. S. Lui. Stochastic analysis and file availability enhancement for bt-like file sharing systems. In *Fourteenth IEEE International Workshop on Quality of Service (IWQoS) 2006*, New Haven, CT, USA, 2006.

[14] R. Gaeta, G. Galbo, S. Bruell, M. Gribaudo, and M. Sereno. A simple analytical framework to analyze search strategies in large-scale peer-to-peer networks. In *Performance Evaluation*, volume 62(1-4), October 2005.

[15] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *Proceedings of IEEE Infocom*, 2005.

[16] R. Graham, D. Knuth, and O. Patashnik. Concrete mathematics, 2nd edition. In *Addison-Wesley*, 1994.

[17] M. Hanckowiak, M. Karonski, and A. Panconesi. A faster distributed algorithm for computing maximal matchings deterministically. In *Proc., 18th ACM Symp. on Principles of Distributed Computing*, 1999.

[18] M. Izal, G. Urvoy-Keller, E. E. Biersack, P. Felber, A. A. Hamra, and L.Garces-Erice. Dissecting bittorrent: Five months in a torrents lifetime. In *PAM*, Antibes Juan-les-Pins, France, Apr 2004.

[19] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should internet service providers fear peer-assisted content distribution? In *ACM IMC*, 2005.

[20] L. Kleinrock. *Queueing Systems. Volume 1: Theory.* John Wiley & Sons, 1975.

[21] L. Kleinrock. *Queueing Systems.* Wiley-Interscience, 1976.

[22] T. Kurtz. *Approximation of Population Processes, Vol. 36.* CBMS-NSF Regional Conf. in Applied Mathematics, 1981.

[23] A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest first and choke algorithms are enough. In *ACM SIGCOMM/USENIX IMC'2006*, October 2006.

[24] M. Lin, B. Fan, J. C. S. Lui, and D. M. Chiu. Stochastic analysis of file swarming systems. In *Performance*, 2007.

[25] T. Ma, C. Lee, J. C. S. Lui, and D. K. Yau. Incentive and service differentiation in P2P networks: A game theoretic approach. In *IEEE/ACM Transactions on Networking*, volume 14(5), 2006.

[26] T. Ma, S. C. Lee, J. C. S. Lui, and D. K. Yau. A game theoretic approach to provide incentive and service differentiation in P2P networks. In *ACM SIGMETRICS/PERFORMANCE*, June 2004.

[27] L. Massoulie and M. Vojnovic. Coupon replication systems. In *Proc. ACM SIGMETRICS*, Banff, Alberta, Canada, 2005.

[28] J. Mundinger, R. Weber, and G. Weiss. Optimal scheduling of peer-to-peer file dissemination. Preprint version in arXiv, 2006.

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

[30] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *Proc. ACM SIGCOMM*, Portland, Oregon, USA, August 2004.

[31] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *Proc. ACM SIGCOMM*, 2004.

[32] I. Reed and G. Solomon. Polynomial codes over certain finite fields. In *Journal of the Society of Industrial and Applied Mathematics*, 1960.

[33] S. Sanghavi, B. Hajeck, and L. Massoulie. Efficient data dissemination in unstructured networks. in submission, 2006.

[34] X. Yang and G. de Veciana. Service capacity of peer to peer networks. In *IEEE INFOCOM*, March 2004.

[35] Y. Zhou, D. M. Chiu, and J. C. S. Lui. A simple model for analyzing P2P streming protocols. In *IEEE International Conference on Network Protocols (ICNP)*, 2007.

Appendix

Proof for Theorem 7: Due to the self scaling property of P2P systems, the service capacity of the system is proportional to the number of peers. Therefore, one can model the P2P file distribution system within this ISP as an $M/D/\infty$ queueing system with arrival rate λ and service time T.

Let p_n denote the probability that there are n peers in the ISP. Since the service time is T, the probability that there are n peers in the ISP is equal to the probability that there are n arrivals between time $[t-T, t]$. Since the number of peers arriving in a time interval of length T is Poisson distributed with mean λT, we immediately obtain

$$p_n = \frac{(\lambda T)^n}{n!} e^{-\lambda T} = \frac{\bar{n}^n}{n!} e^{-\bar{n}} \qquad n = 0, 1, \ldots .$$

The above statement is valid for all $t > T$, and thus also for the limiting distribution.

Now consider when these peers have to download content from external peers, e.g., peers which belong to *other ISPs*. Assume that there are n peers within this ISP at a certain time. Let v_i denote the fraction of file content that peer i has obtained so far. Since the size of the file is 1, we have $v_i < 1$ for $i \in \{1, \ldots, n\}$. If $v = \sum_{i=1}^{n} v_i < 1$, then these n peers need to download *at least* $(1 - v)$ fraction of the file content from external peers before the next peer departure from this ISP.

We use the method of the *imbedded Markov Chain* [20] and select the departure points as our observation points. Since the arrival is a Poisson process, we have

$$p_n = \text{Prob}\,(\text{departure leaves } n \text{ peers in the systems})\,.$$

When a peer departs and observes that there are n peers within this ISP with $v = \sum_{i=1}^{n} v_i < 1$, then this ISP needs to consume at lease $(1 - v)$ of incoming cross-ISP traffic before the next peer departure.

When there are exactly n arrivals from a Poisson process in $[0, t]$, the unordered arrival times are uniformly, independently distributed over $[0, t]$. In our system, it means that all these n downloading progresses are uniformly, independently distributed over $[0, 1]$. Formally, let X_i be the random variable denoting v_i, we have $X_i \sim U[0, 1], i = 1, \ldots, n$. We

are interested in $Y_n = \sum_{i=1}^{n} X_i$ and its corresponding density function $f(v|n)$. To derive Y_n and $f(v|n)$, one can use Laplace transformation method:

$$\begin{aligned} X_i(s) &= \frac{1}{s}\left(1 - e^{-s}\right) \\ Y_n(s) &= \prod_{i=1}^{n} X_i(s) = \frac{1}{s^n}(1 - e^{-s})^n = \frac{1}{s^n}\sum_{j=0}^{n} C_n^j (-1)^j e^{-js}. \end{aligned}$$

Thus

$$\begin{aligned} f(v|n) &= \sum_{j=0}^{n} C_n^j (-1)^j \frac{(v-j)^{n-1}}{(n-1)!} u(v-j) \\ &= \sum_{j=0}^{\lfloor v \rfloor} C_n^j (-1)^j \frac{(v-j)^{n-1}}{(n-1)!} \end{aligned}$$

Focusing on the range $0 \le v < 1$, we have

$$f(v|n) = \frac{v^{n-1}}{(n-1)!}, \qquad 0 \le v < 1, n = 1, 2, \ldots.$$

Let d denote the incoming cross-ISP traffic between two consecutive peers departures. Since these n peers need to download at least $(1-v)$ fraction of the file through the cross-ISP link before the next peer departure, we have

$$E(d|n) \ge \int_0^1 (1-v) f(v|n) dv = \frac{1}{(n+1)!}, \qquad n = 1, 2, \ldots.$$

Now consider the case $n = 0$. When a departing peer observes that there is no peer in the ISP, this means that new arriving peers need to download exactly one copy of the file via the cross-ISP link before the next peer departure. Thus

$$E(d|0) = 1 = \frac{1}{(0+1)!}.$$

Given $E(d|n)$ and p_n, one can derive $E(d)$, the average cross-ISP traffic caused by each departure.

$$\begin{aligned} E(d) &= E(E(d|n)) = \sum_{n=0}^{\infty} p_n E(d|n) \\ &\ge \sum_{n=0}^{\infty} \frac{\bar{n}^n}{n!} e^{-\bar{n}} \frac{1}{(n+1)!} = e^{-\bar{n}} \bar{n}^{-1/2} I_1(2\sqrt{\bar{n}}) \end{aligned}$$

where $I_1(x)$ is the *modified Bessel function.* ∎

Proof for Theorem 8: Similar to the $M/D/\infty$ formulation in the proof of Theorem 7, one can use the method of the *Imbedded Markov Chain* and select the departure points as the observing points.

Consider the situation that a peer departs and observes that there are n peers within this ISP. The progress of these n peers are uniformly and independently distributed over $[0,1]$, i.e., $X_i \sim U[0,1], i = 1, \ldots, n$. Consider the peer whose progress is maximal. According to the ELP, those content held by this peer would not generate any cross-ISP traffic before the next peer departure. On the other hand, those content that are not being held by this peer may or may not cause a data transfer over the cross-ISP link before the next peer departure (the content may be held by other internal peers). To derive the upper bound, we ignore the collision of two or more peers request the same chunk from external peers at the same time. We consider $Z_n = \max_{i=1}^{n} X_i$ and its corresponding density function as $g(v|n)$.

Since $\text{Prob}(Z_n \leq v) = \prod_{i=1}^{n} \text{Prob}(X_i \leq v)$, we have

$$g(v|n) = nv^{n-1}, \qquad 0 \leq v \leq 1, n = 1, 2, \ldots$$

This requires at most $(1 - v)$ fraction of the file via the cross-ISP link before the next peer departure. We have

$$E(d|n) \leq \int_0^1 (1-v)g(v|n)dv = \frac{1}{n+1}, \qquad n = 1, 2, \ldots$$

Consider the case that $n = 0$. When a departing peer observes that there are no peer in the ISP, the new arriving peers need to download one copy of the file via the cross-ISP link before the next peer departure. Thus

$$E(d|0) = 1 = \frac{1}{0+1}$$

Given the upper bound of $E(d|n)$ and p_n, one can derive the upper bound of $E(d)$, the average cross-ISP traffic caused by each departure.

$$\begin{aligned} E(d) &= E(E(d|n)) = \sum_{n=0}^{\infty} p_n E(d|n) \\ &\leq \sum_{n=0}^{\infty} \frac{\bar{n}^n}{n!} e^{-\bar{n}} \frac{1}{n+1} = \frac{1}{\bar{n}}(1 - e^{-\bar{n}}) \end{aligned}$$

∎

Proof for Theorem 9: Let T denote the random variable of the file downloading time of peers. Suppose T is a discrete random variable with possible outcomes of $\tau_1, \tau_2, \ldots, \tau_m$ and

$$\text{Prob}(T = \tau_i) = q_i, \quad i = 1, 2, \ldots, m, \text{ and } \sum_{i=1}^{m} q_i = 1.$$

Let us first derive p_n, the probability that there are n peers in the ISP. One can split the Poisson arrival with rate λ into m independent Poisson arrival streams. The arrival rate of peers with downloading time τ_i is denoted by λ_i. Thus

$$\lambda_i = \lambda q_i, \qquad i = 1, 2, \ldots, m.$$

Using similar argument as in the previous section, the number of peers with downloading time τ_i in the ISP is Poisson distributed with mean $\lambda_i \tau_i$, therefore the probability that we have n peers of downloading time τ_i is

$$p_{n,i} = \frac{(\lambda_i \tau_i)^n}{n!} e^{-\lambda_i \tau_i} = \frac{\bar{n}_i^n}{n!} e^{-\bar{n}_i}, \quad i = 1, 2, \ldots, m, \; n = 0, 1, \ldots$$

where $\bar{n}_i = \lambda_i \tau_i = \lambda q_i \tau_i$. The number of peers with downloading time τ_i is independent of the number of peers with other downloading time in the ISP. Since the sum of independent Poisson random variables is again Poisson, it follows that the total number of peers in the ISP p_n is Poisson distributed.

$$p_n = \frac{\bar{n}^n}{n!} e^{-\bar{n}}, \quad n = 0, 1, \ldots$$

where $\bar{n} = \sum_{i=1}^{m} \bar{n}_i = \lambda \sum_{i=1}^{m} q_i \tau_i$.

Assume that there are n peers in the ISP at a given time. Similar to the homogenous case analysis, we know that the content that peer i holds, denoted as v_i, is uniformly and independently distributed over $[0, 1]$. Let X_i be the random variable denoting v_i, we have $X_i \sim U[0, 1], i = 1, \ldots, n$. Given p_n and X_i, one can derive the lower bound and upper bound of the cross-ISP traffic similar to Theorem 7 and Theorem 8. The result is

$$e^{-\bar{n}} \bar{n}^{-1/2} I_1(2\sqrt{\bar{n}}) \leq E(d) \leq \frac{1}{\bar{n}}(1 - e^{-\bar{n}}),$$

where $\bar{n} = \lambda \sum_{i=1}^{m} q_i \tau_i$ and $I_1(x)$ is the modified Bessel function.

In fact, since each distribution function can be approximated arbitrary close by a discrete distribution function, one can conclude that the result holds for general downloading time distribution. ∎

Proof for Theorem 10: Since there is no file content within the ISP at time $t = 0$, peers in this ISP should download at least one copy of the file through cross-ISP link. Thus the average cross-ISP traffic generated by each peer, denoted by $E(d)$, satisfies

$$E(d) \geq \frac{1}{n}.$$

To derive the upper bound of the average cross-ISP traffic, similar to the analysis in the regular arrival case, suppose the downloading time T is a discrete random variable. Its possible values are $\tau_1, \tau_2, \ldots, \tau_m$, Without loss of generality, we assume that $\tau_1 \leq \tau_2 \leq \ldots \leq \tau_m$. Peers arrive to the ISP at the same time $t = 0$, and peers may depart the system at time $t = \tau_i$, $i = 1, 2, \ldots, m$.

Let d_i denote the incoming cross-ISP traffic during the time interval $[\tau_{i-1}, \tau_i]$ ($[0, \tau_1]$ for d_1). Let D denote the total incoming cross-ISP traffic during the whole flash crowd downloading, we have $D = \sum_{i=1}^{m} d_i$.

Consider those peers which depart at $t = \tau_1$. The incoming cross-ISP traffic generated during $[0, \tau_1]$ is d_1, which is one copy of the file. After the

departure of peers at τ_1, the maximal progress of downloading in the ISP at time τ_1 are those peers who will finish at τ_2, their progress at this time is τ_1/τ_2. Thus during $[\tau_1, \tau_2]$, the internal peers will at most download $(1 - \tau_1/\tau_2)$ of the file context from external peers, i.e. $d_2 \leq 1 - \tau_1/\tau_2$. Similarly, one can consider the time interval $[\tau_{i-1}, \tau_i]$, the cross-ISP traffic $d_i \leq 1 - \tau_{i-1}/\tau_i$. Therefore, the total cross-ISP traffic during $[0, \tau_m]$ is

$$\begin{aligned} D &= \sum_{i=1}^{m} d_i \leq 1 + \sum_{i=2}^{m} (1 - \frac{\tau_{i-1}}{\tau_i}) = m - \sum_{i=2}^{m} \frac{\tau_{i-1}}{\tau_i} \\ &\leq m - (m-1)\left(\frac{\tau_1}{\tau_m}\right)^{1/(m-1)} \qquad m \geq 2. \end{aligned}$$

The function $y = x - (x-1)a^{1/(x-1)}$ is an increasing function of x when $x \geq 2$, $0 < a \leq 1$. Thus

$$D \leq \lim_{m \to \infty} m - (m-1)\left(\frac{\tau_1}{\tau_m}\right)^{1/(m-1)} = 1 + \log\left(\frac{\tau_m}{\tau_1}\right).$$

Since there are n peers, the average cross-ISP traffic each peer generated, denoted by $E(d)$, satisfies

$$E(d) \leq \left(1 + \log\left(\frac{\tau_{max}}{\tau_{min}}\right)\right)/n. \qquad \blacksquare$$