

# Stochastic Analysis of File Swarming Systems

Minghong Lin<sup>a</sup>, Bin Fan<sup>a</sup>, John C.S. Lui<sup>a</sup>, Dah-Ming Chiu<sup>b</sup>

<sup>a</sup>*Department of Computer Science and Engineering, {mhlín, bfan, cslui}@cse.cuhk.edu.hk*

<sup>b</sup>*Department of Information Engineering, dmchiu@ie.cuhk.edu.hk  
The Chinese University of Hong Kong, Shatin, Hong Kong*

---

## Abstract

File swarming (or file sharing) is one of the most important applications in P2P networks. In this paper, we propose a stochastic framework to analyze a file swarming system under realistic setting: constraints in upload/download capacity, collaboration among peers and incentive for chunk exchange. We first extend the results in the coupon system [18] by providing a tighter performance bound. Then we generalize the coupon system by considering peers with limited upload and download capacity. We illustrate the *last-piece problem* and show the effectiveness of using forward error-correction (FEC) code and/or multiple requests to improve the performance. Lastly, we propose a framework to analyze an incentive-based file swarming system. The stochastic framework we propose can serve as a basis for other researchers to analyze and design more advanced features of file swarming systems.

*Key words:* Performance Modeling; BitTorrent; P2P File Sharing

---

## 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 swarming (or file sharing) is one of the most important applications in P2P networks. In general, a file swarming 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 *chunks*. Each peer can get these chunks either from a server, or from other peers holding those chunks that it does not already have. Each peer offers upload service to other peers, and in return, each peer tries to obtain a missing chunk 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, file swarming 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 swarming application is implemented in peer-to-peer file sharing networks such as eDonkey, KaZaA, and it is the core functionality of the popular BitTorrent (BT) [1] protocol.

The work by the authors in [24] suggest that file swarming 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 [21] 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 [18] 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 [24], and that the system performs well under the flash crowd scenario, even when the rarest first chunk selection policy is replaced by some random coupon selection policies. However, strong assumptions are made in [18], in particular, the authors assume that peers have infinite upload capacity (or relatively large as compare with the download capacity).

The aim of this paper is to provide a deeper understanding to the file swarming 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 swarming system, and we investigate the performance of the system under constraints on upload capacity, download capacity, peer selection policies (including random chunk selection and coordinated matching). The contributions of our work are

- We generalize some of the results in the coupon system [18] 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) [22] coding technique for file sharing.
- We relax the unlimited upload capacity assumption in [18], 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 chunk 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.

The balance of this paper is as follows. In Section 2, we present a generic model for a file swarming system. In Section 3, we present an analytical model of an altruistic file swarming system wherein each peer has unlimited amount (or sufficiently large amount) of upload capacity and derive performance measure such as the average file downloading time. In Section 4, we present the model of an altruistic file swarming system with limited upload and download capacity. In Section 5, an analytical model of an incentive-based file swarming system is presented and we derive various important performance measures. Extensive simulations and the related results are given in Section 6. Related work is given in Section 7 and Section 8 concludes.

## 2 Model Description

Let us consider a peer-to-peer file swarming 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  $\mathcal{F} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \dots \cup \mathcal{C}_K$ , 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 swarming 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  $\lambda$ . 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<sup>1</sup>. This initial chunk is randomly chosen by the server with equal probability  $1/K$  for chunks  $\mathcal{C}_1 \dots \mathcal{C}_K$ . When a peer finishes downloading all  $K$  chunks, the peer will depart immediately.

Similar to [18], we assume that this P2P file swarming 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

---

<sup>1</sup> This assumption is similar to the one made in [18]

of a bitmap is negligible compared to the transfer time of a chunk. Let 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  $C_k \subset \mathcal{F}_B \setminus \mathcal{F}_A$ ). 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 [4]. 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.

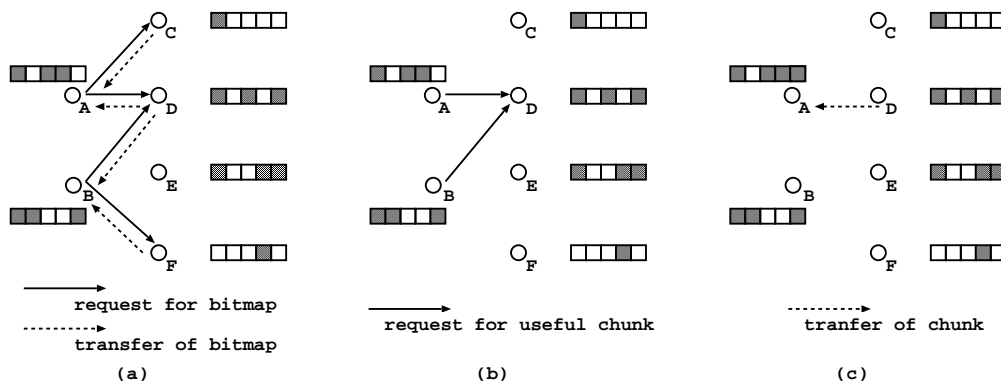


Fig. 1. A simple illustration of a transfer dynamic within one time slot with  $\mathcal{F} = \mathcal{C}_1 \cup \mathcal{C}_2 \cdots \cup \mathcal{C}_5$

Figure 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  $C_1, C_3$  and  $C_4$ . In Figure 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 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 1(c) shows that peer  $D$  transfers  $C_5$  which is requested by  $A$ . At the end of a time slot, peer  $A$  obtains  $C_5$  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 [18]. 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 [18]. 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 swarming 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 swarming 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 swarming systems. In later sections, we will derive the performance of such system, and show why and how it can achieve good performance.

### 3 Altruistic File Swarming System with Constraint in Download Capacity

In this section, we consider the file swarming 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 [18].

### 3.1 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). 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  $i = 1, 2, \dots, K - 1$  ( $i \neq K$  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

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

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  $q_i(t) = \sum_{j=1}^{K-1} p_{i,j} y_j(t) / y(t)$ ,  $i = 1, 2, \dots, K - 1$ .

Given the system state  $\mathbf{Y}(t) = \{y_i(t)\}_{i \in \{1, \dots, K-1\}}$ , it is easy to verify that  $(\mathbf{Y}(t))_{t \geq 0}$  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  $i \in \{1, \dots, K - 1\}$ ,

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

We analyze the system under a large population asymptotic regime. Note that this is a *density dependent jump Markov process* [14]. It converges to the solution of the differential equations

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

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

### 3.2 Steady State Analysis

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

#### 3.2.1 Altruistic File Swarming Without FEC

In this section we focus on the steady state performance and its *equilibrium point*. An equilibrium point is the point  $\hat{\mathbf{Y}} = (y_1, y_2, \dots, y_{K-1})$  such that if  $\mathbf{Y}(t) = \hat{\mathbf{Y}}$ , then  $\mathbf{Y}(t') = \hat{\mathbf{Y}}$  for all  $t' \geq t$ . The necessary and sufficient condition for  $\hat{\mathbf{Y}}$  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), we have the following equations at the equilibrium point  $\hat{\mathbf{Y}}$ :  $\lambda = y_i(1 - (1 - q_i)^m)$ ,  $i = 1, 2, \dots, K-1$ .

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  $\hat{\mathbf{Y}} = (y_1, \dots, y_{K-1})$  by using Little's theorem [13]:  $\lambda T_i = y_i$ . Define  $T = \sum_{j=1}^{K-1} T_j$  as the average file downloading time in the P2P file swarming system, we have  $y_i/y = T_i/T$ . Finally, one can obtain the following equations at the equilibrium point  $\hat{\mathbf{Y}}$ :

$$T_i = \frac{1}{1 - (1 - q_i)^m} \quad \text{and} \quad q_i = \sum_{j=1}^{K-1} \frac{T_j}{T} p_{i,j}, \quad \text{for } i = 1, 2, \dots, K-1, \quad (3)$$

One can observe that  $T_i$  of Eq. (3) does not depend on  $\lambda$ . So even when the arrival rate  $\lambda$  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 file swarming 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.  $1 \leq T_1 < T_2 < \dots < T_{K-1}$ .*

**Proof:** According to Eq. (3) we have  $q_i \leq 1$ . Therefore, one can conclude that  $T_i \geq 1$  for  $i = 1, \dots, K-1$ . According to Eq. (1), when  $i > i'$ ,  $p_{i,j} \leq p_{i',j}$  holds for  $j = 1, \dots, K-1$  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  $K^{\text{th}}$  harmonic number.

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

$$\begin{aligned} q_i &> \frac{1}{K-1} \left( \sum_{j=1}^{K-1} \frac{T_j}{T} \right) \left( \sum_{j=1}^{K-1} p_{i,j} \right) = \frac{1}{K-1} \left( K-1 - \sum_{j=1}^i \frac{C_i^j}{C_K^j} \right) \\ &= 1 - \left( \frac{1}{K-1} \right) \left( \frac{i}{K-i+1} \right) \quad (\text{"Concrete Mathematics" [10], p174.}) \end{aligned}$$

One can apply it to Eq. (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

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

We can apply it to Eq. (3) to obtain an upper bound of  $q_i$  as

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

With this upper bound of  $q_i$ , one can substitute it to Eq. (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 swarming 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(a) and Fig. 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

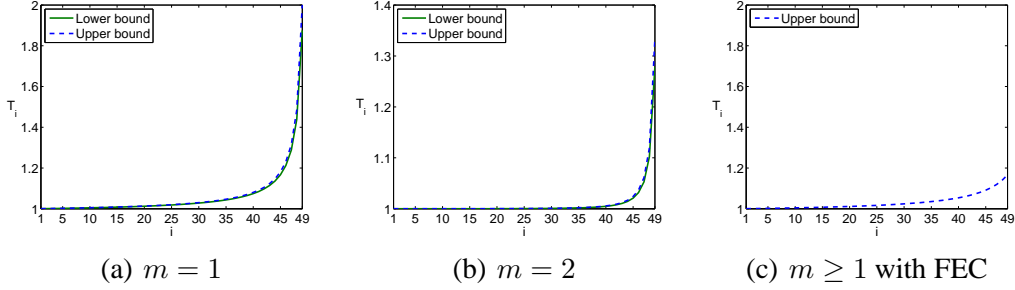


Fig. 2. Illustration on the last-piece problem: bounds of  $T_i$  for  $m = 1, 2$  and  $m \geq 1$  with FEC.  $K = 50$  chunks

measure of the average file downloading time  $T$ . Secondly, one can observe that the sojourn times  $T_i$  are very close to 1 for  $i \ll K - 1$ , but when  $i$  approaches  $K - 1$ , Fig. 2(a) (and Fig. 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(a)), it costs 2 time slots on average to download the last chunk but when  $m = 2$  (as shown in Fig. 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

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

Combining the upper and lower bounds, Theorem 1 can be shown as claimed. ■

**Remark:** Note that when  $m = 1$ , the system corresponds to the “open and flat” case of the coupon system [18], 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  $K^{\text{th}}$  harmonic number,  $H_K = \log K + \gamma + O(K^{-1})$ , where  $\gamma = 0.5772\dots$  is the *Euler-Mascheroni constant*. Thus  $T = K + \log K + O(1)$ . Therefore, we obtain a tighter bound than [18].

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:**

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

**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.

### 3.2.2 Altruistic File Swarming 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 [22]. 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

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

**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  $\alpha$ ,  $T_{FEC}$  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. [9] 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(a) and Fig. 2(b) shown. 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(c). This bound holds for all  $m \geq 1$ . From Fig. 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.

#### 4 Altruistic File Swarming System with Download and Upload Constraints

In this section, we consider the file swarming 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 3 and the coupon replication system [18]. 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.

#### 4.1 Model Formulation

As in Section 3, 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  $R \sim \text{Bernoulli}(y-2, \frac{1}{y-1})$ . 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  $k \in \{0, 1, \dots\}$ .

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

$$\mathbf{Y} \longrightarrow \mathbf{Y} + e_1 \quad \text{with rate } \lambda,$$

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

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

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

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

#### 4.2 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:

$$\frac{1}{T_i} = \sum_{j=1}^{K-1} \left( \frac{T_j}{T} p_{i,j} \sum_{r=0}^{\infty} \frac{e^{-1}}{r!} \alpha_{j,r} \right), \quad i = 1, 2, \dots, K-1 \quad (4)$$

where

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

In Section 3, we have shown that a file swarming 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. (4), we use the same technique in proving 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 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 3, 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

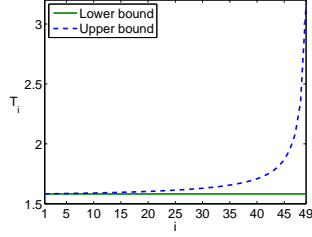


Fig. 3. Numerical results illustrated for the bounds of  $T_i$  for  $m = 1$  when  $K = 50$

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

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

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 3.2.2, 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.  $\blacksquare$

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.

## 5 Incentive File Swarming 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 swarming 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.

### 5.1 Without Incentive Mechanism

We assume at the beginning of each time slot, every peer will run some distributed *maximal matching algorithm* [11], 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

$$\frac{dy_i(t)}{dt} = \begin{cases} \lambda - y_1 \sum_{j=1}^{K-1} \frac{y_j(t)}{y(t)} p_{1,j} & i = 1 \\ y_{i-1}(t) \sum_{j=1}^{K-1} \frac{y_j(t)}{y(t)} p_{i-1,j} - y_i(t) \sum_{j=1}^{K-1} \frac{y_j(t)}{y(t)} p_{i,j} & i = 2, \dots, K-1. \end{cases} \quad (5)$$

One can find that, Eq. (5) is equivalent to the differential equations given in Eq. (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 4. 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.

## 5.2 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

$$p'_{i,j} = \begin{cases} 1 - \frac{C_i^j}{C_K^j} & 1 \leq j \leq i \leq K-1, \\ 1 - \frac{C_i^i}{C_K^i} & 1 \leq i < j \leq K-1. \end{cases} \quad (6)$$

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

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

To simplify our notation, let us denote  $w_{i,j} = p'_{i,j} + p'_{i,K-j}$  ( $i, j = 1, \dots, K - 1$ ). 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

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

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  $T_i = T_{K-i}$ .  $\blacksquare$

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

$$\frac{1}{T_i} = \sum_{j=1}^{K-1} \frac{T_j}{T} p'_{i,j}, \quad i = 1, 2, \dots, K-1, \quad (7)$$

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

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

**Proof:** Let  $1 \leq i' < i \leq \lfloor K/2 \rfloor$ . Base 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

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

**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

$$\begin{aligned} \frac{1}{T_i} &= \sum_{j=1}^{K-1} \frac{T_j}{T} p'_{i,j} = \frac{1}{2} \sum_{j=1}^{K-1} \frac{T_j}{T} w_{i,j} \leq \frac{1}{K-1} \sum_{j=1}^{K-1} \frac{T_j}{T} \cdot \sum_{j=1}^{K-1} p'_{i,j} = \frac{1}{K-1} \sum_{j=1}^{K-1} p'_{i,j} \\ &= \frac{1}{K-1} \left[ \sum_{j=1}^i \left(1 - \frac{C_i^j}{C_K^j}\right) + \sum_{j=i+1}^{K-1} \left(1 - \frac{C_j^i}{C_K^i}\right) \right] \\ &= 1 - \frac{1}{K-1} \left[ \frac{i}{K-i+1} + \frac{K-i}{i+1} - \frac{1}{C_K^i} \right]. \end{aligned}$$

Therefore, we obtain the lower bound of  $T$  as

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

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

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

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

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

Thus the upper bound of  $T$  is

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

**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 swarming 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.

## 6 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 swarming 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 3 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 [18].

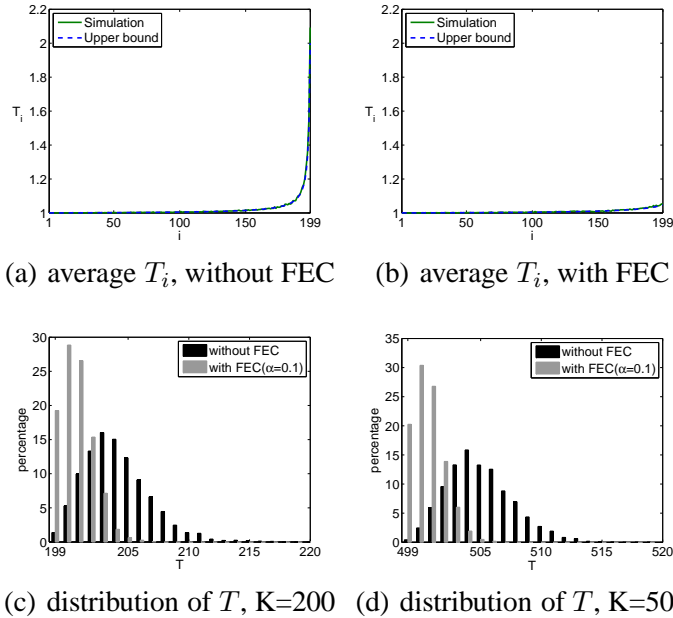


Fig. 4.  $T_i$  and  $T$  for  $m = 1$ , constraint on download capacity only

Fig. 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<sup>2</sup>. This indicates that our analytical result is very accurate. Fig. 4(b) 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. 4(c)-(d) illustrate the probability density function for the average file downloading time  $T$ , with and without using FEC, for  $K = 200$  and

<sup>2</sup> For the analytical results, since the spread of the bounds is very tight, we simply plot the upper bound of  $T_i$

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 3 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.5(a) presents the average sojourn time  $T_i$  without FEC. The simulation results are similar to the analytical results again. Comparing Fig. 5(a) and Fig. 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. 5(b). Notice that we only give out a loose upper bound of  $T_i$  in Fig. 5(b), which is also the upper bound of the system without FEC in Fig. 5(a). Now we examine the probability density function of  $T$  in Fig. 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 4. We consider  $m = 1$  in our analysis thus we set  $m = 1$  in this simulation. Fig.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. 6(a) and Fig. 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 problem. Also note that when we have upload and download capacity constraints, the variance on  $T$  is significant larger than the previous experiments. This can be confirmed by Fig. 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.

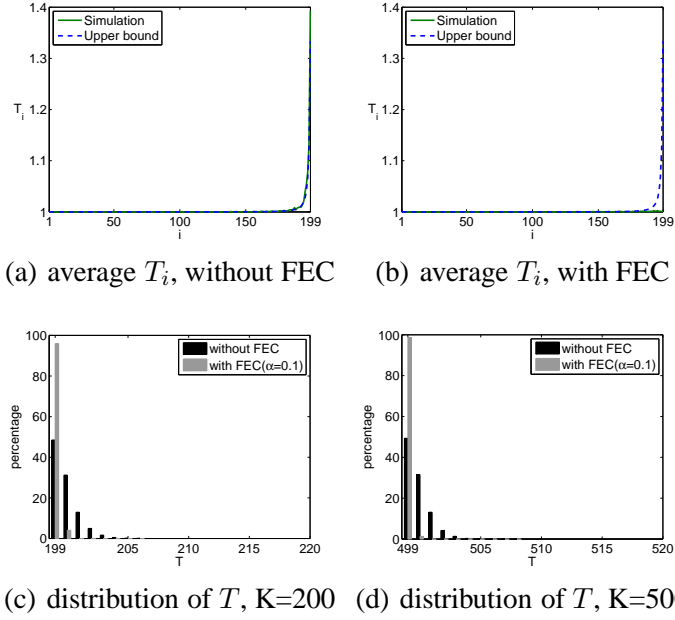


Fig. 5.  $T_i$  and  $T$  for  $m = 2$ , constraint on download capacity only

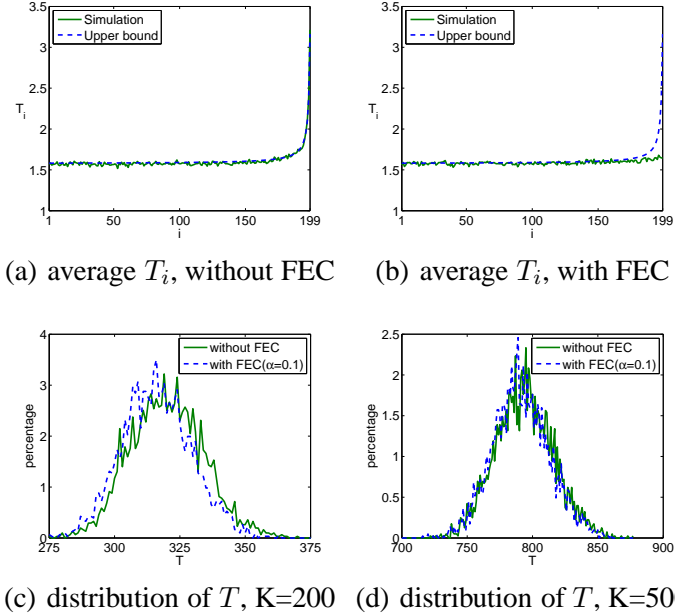


Fig. 6.  $T_i$  and  $T$  for  $m = 1$ , constraint on upload and download capacity

**Experiment 4:** This experiment is to validate the coordinated matching system with incentive mechanism as described in Section 5. Fig.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*<sup>3</sup> in our analytical bound and sim-

<sup>3</sup> This problem is reported as first block problem in [15] by measurement study as the slow startup due to choking.

ulation 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 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. 7(b) has indicated. From Fig. 7(c) and 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 the 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.

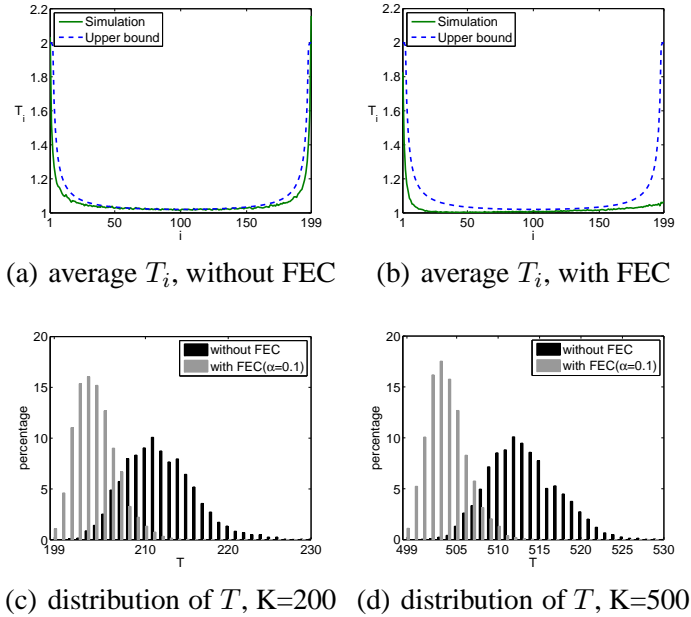


Fig. 7.  $T_i$  and  $T$  for coordinated matching, incentive mechanism, with constraint on upload and download capacity

## 7 Related Work

There are numerous empirical studies on the BT protocol, for instance, [2, 5, 12, 15, 20]. Izal et al. [12] present the traffic information on peers behavior collected during a five-month period. Pouwelse et al. [20] 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. [5] 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. [15] 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. [2] 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. [24] 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. [21] extend the coarse-grain Markovian model in [24] 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 [21, 24] 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 [21, 24]. Fan et al. [7] 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. [19] propose a deterministic scheduling algorithm to achieve the optimal makespan which requires global knowledge. Sanghavi et al. [23] also propose a gossip-like randomized algorithm requiring only local knowledge. Both studies in [19] and [23] 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 [18]. 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 paper improves and extends the result in [18] 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. [8] 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 [3, 16, 17], the authors present a mathematical analysis on service differentiation in resource allocation for P2P networks. In [6], the authors present a mathematical framework to study the tradeoff between performance and fairness in BT-like systems. In [25], authors present the first analytical model of BT-like sys-

tems and quantify the tradeoff between scalability and QoS support for multimedia streaming applications.

## 8 Conclusion

In this paper, we propose a probabilistic model which generalizes the model in [18] to capture the basic properties of a file swarming system. Under the same assumption as [18](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.

## References

- [1] Bittorrent protocol. <http://www.bittorrent.com/protocol.html>.
- [2] R. Bindal and P. Cao. Can self-organizing P2P file distribution provide qos guarantees? In *OSR Special Issue on Self-Organizing Systems*, 2006.
- [3] 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.
- [4] B. Cohen. Incentives build robustness in bittorrent. <http://www.bittorrent.org/bittorrentecon.pdf>, May 2003.
- [5] 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.
- [6] B. Fan, D.-M. Chiu, and J. C. S. Lui. The delicate tradeoffs in bittorrent-like file sharing protocol design. In *ICNP, 2006*.

- [7] 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.
- [8] 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.
- [9] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *Proceedings of IEEE Infocom*, 2005.
- [10] R. Graham, D. Knuth, and O. Patashnik. Concrete mathematics, 2nd edition. In *Addison-Wesley*, 1994.
- [11] 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.
- [12] 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.
- [13] L. Kleinrock. *Queueing Systems*. Wiley-Interscience, 1976.
- [14] T. Kurtz. *Approximation of Population Processes, Vol. 36*. CBMS-NSF Regional Conf. in Applied Mathematics, 1981.
- [15] A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest first and choke algorithms are enough. In *ACM SIGCOMM/USENIX IMC'2006*, October 2006.
- [16] 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.
- [17] 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.
- [18] L. Massoulié and M. Vojnovic. Coupon replication systems. In *Proc. ACM SIGMETRICS*, Banff, Alberta, Canada, 2005.
- [19] J. Munding, R. Weber, and G. Weiss. Optimal scheduling of peer-to-peer file dissemination. Preprint version in arXiv, 2006.
- [20] 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.
- [21] 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.
- [22] I. Reed and G. Solomon. Polynomial codes over certain finite fields. In *Journal of the Society of Industrial and Applied Mathematics*, 1960.

- [23] S. Sanghavi, B. Hajeck, and L. Massoulié. Efficient data dissemination in unstructured networks. in submission, 2006.
- [24] X. Yang and G. de Veciana. Service capacity of peer to peer networks. In *IEEE INFOCOM*, March 2004.
- [25] Y. Zhou, D. M. Chiu, and J. C. S. Lui. A simple model for analyzing P2P streaming protocols. In *IEEE International Conference on Network Protocols (ICNP)*, 2007.

**Minghong Lin** received his B.S. degree in Computer Science from University of Science and Technology of China in 2006. Currently, he is an M.Phil student in the Department of Computer Science and Engineering in the Chinese University of Hong Kong. His current interests lie in modeling and analysis of computer networks.

**Bin Fan** received his B.S. degree in Computer Science from University of Science and Technology of China, and his M.Phil degree in Computer Science from the Chinese University of Hong Kong. Currently he is a research assistant in the Chinese University of Hong Kong. He is a student member of IEEE.

**John C.S. Lui** received his Ph.D. in Computer Science from UCLA. He worked in the IBM San Jose/Almaden Research Laboratory before joining the Chinese University of Hong Kong. Currently, he is the chair of the Computer Science & Engineering Department at CUHK and he leads the Advanced Networking & System Research Group. His research interests span both in systems as well as in theory/mathematics in computer communication systems. John received various departmental teaching awards and the CUHK Vice-Chancellor's Exemplary Teaching Award, as well as the co-recipient of the Best Student Paper Awards in the IFIP WG 7.3 Performance 2005 and the IEEE/IFIP Network Operations and Management (NOMS) Conference.

**Dah-Ming Chiu** received the B.Sc degree in Electrical Engineering from Imperial College, University of London, and the Ph.D degree from Harvard University, in 1975 and 1980 respectively. He was a Member of Technical Staff with Bell Labs from 1979 to 1980. From 1980 to 1996, he was a Principal Engineer, and later a Consulting Engineer at Digital Equipment Corporation. From 1996 to 2002, he was with Sun Microsystems Research Labs. Currently, he is a professor in the Department of Information Engineering in The Chinese University of Hong Kong. He is known for his contribution in studying network congestion control as a resource allocation problem, the fairness index, and analyzing a distributed algorithm (AIMD) that became the basis for the congestion control algorithm in the Internet. His current research interests include economic issues in networking, P2P networks, network traffic monitoring and analysis, and resource allocation and congestion control for the Internet with expanding services. Two recent papers he co-authored with students have won best student paper awards from the IFIP Performance Conference and the IEEE NOMS Conference. Recently, Dr Chiu has served on the TPC of IEEE Infocom, IWQoS and various other conferences. He is a member of the editorial board of the IEEE/ACM Transactions on Networking, and the International Journal of Communication Systems (Wiley).