PAPER

Analysis of Optimal Scheduling in Tit-for-Tat-based P2P File Distribution

Masashi HASEGAWA[†a)], *Nonmember*, Masahiro SASABE[††b)], *and* Tetsuya TAKINE[†c)], *Members*

SUMMARY Peer-to-Peer (P2P) file distribution systems can efficiently disseminate massive contents, such as disk images of operating systems, from a server to many users in a piece-by-piece manner. In particular, the BitTorrent protocol optimizes each peer's download speed by applying the tit-for-tat (TFT) strategy, where each peer preferentially uploads piece(s) to peer(s) from which it can download missing pieces faster. To the best of our knowledge, however, the optimality of TFT-based P2P file distribution has not been studied sufficiently. In this paper, we aim to understand the optimal scheduling in TFT-based P2P file distribution. First, we develop a discrete-time model of TFT-based P2P file distribution and formulate its optimal scheduling as a two-step integer linear programming problem. The first step is to minimize the average file retrieval time among peers, and the second step is to improve fairness among peers. We analyze the optimal solution obtained by the existing solver and reveal the characteristics of the optimal scheduling. Specifically, we show that it is crucial to distribute pieces from the server indirectly to peers with large upload capacity via those with small upload capacity.

key words: *P2P file distribution, tit-for-tat strategy, analysis of optimal scheduling, integer linear programming (ILP)*

1. Introduction

Distribution of massive contents, e.g., disk images of operating systems (Linux, FreeBSD, etc.) and virtual machines, in the Internet has been attracting many users. The conventional client-server architecture, where a small number of servers serve the contents to clients, potentially has a drawback of scalability to the number of clients, due to the bottleneck of the upload capacity of the servers. P2P (Peer-to-Peer) file distribution systems have been expected to solve such scalability problems [1].

In P2P file distribution systems, a file is divided into small fragments called pieces. Clients not only retrieve those pieces from servers but also transfers them to other clients. Because clients also fill the role of servers, they are called peers. Compared with the client-server architecture, the P2P architecture is scalable to the number of peers because it can utilize the upload capacity of peers in addition to that of servers. This is attractive to the content distributors because they do not need to reinforce the upload capacity of their own servers.

In P2P file distribution systems, however, there still remains a significant problem: How to encourage peers to transfer pieces to other peers. Because the transfer of large files requires a high-rate and/or long-term communication, peers may suffer from the deterioration of their quality of other network services, which they want to enjoy at the same time. As a result, it was pointed out that many peers are negative about uploading pieces they retrieved to other peers in actual systems [2].

To tackle this problem, the Tit-for-Tat (TFT) strategy in game theory is known to be effective [3]. Under the TFT strategy, a player takes a cooperative action initially and then mimics the action taken by the opponent. Therefore, unless the opponent defects, the cooperative relationship between them is sustained. BitTorrent [1] first introduced the TFT strategy into P2P file distribution systems. Note that in P2P file distribution systems, cooperation corresponds to the transfer of pieces to other peers in response to their requests and noncooperation corresponds to the refusal to respond other peers' requests. As a result, each peer has to upload pieces to others so as to retrieve his/her demanded files from them.

There are many studies on examining the effectiveness of the TFT strategy in BitTorrent: Mathematical analyses [4–6], simulation-based approaches [3, 7], and measurement-based approaches [8]. Most of those studies discuss the performance of the original mechanism of BitTorrent and its improvement. To the best of our knowledge, the optimality of such TFT-based P2P file distribution has not been studied sufficiently.

In this paper, we analyze an optimal scheduling in TFT-based P2P file distribution. Specifically, we develop a discrete-time model of TFT-based P2P file distribution and formulate its optimal scheduling as a two-step integer linear programming (ILP). In the two-step ILP, we first minimize the average file retrieving time among peers, and then improve fairness among

Manuscript received March 5, 2014.
Manuscript revised June 3, 2014.

[†]The authors are in the Department of Information and Communications Technology, Graduate School of Engineering, Osaka University, Suita, Osaka, 565-0871, Japan.
[††]The author is in the Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma, Nara, 630-0192, Japan.
a) E-mail: m-hasegawa@post.comm.eng.osaka-u.ac.jp
b) E-mail: sasabe@is.naist.jp
c) E-mail: takine@comm.eng.osaka-u.ac.jp

DOI: 10.1587/transcom.E0.B.1

them under a constraint that the average file retrieving time is minimum. We analyze the optimal solution, which is obtained by an existing solver, and reveal the characteristics of the optimal scheduling.

The rest of the paper is organized as follows. In section 2, we review the related work. After explaining TFT-based P2P file distribution in section 3, section 4 formulates an optimal scheduling in TFT-based P2P file distribution as a two-step ILP. We show some numerical results of the optimal scheduling in section 5 and analyze the mechanism yielding the optimal scheduling in section 6. Finally, we conclude this paper in section 7.

2. Related Work

The evaluation of TFT-based P2P file distribution has been conducted mainly on the BitTorrent system and most of those are summarized in [9]. The TFT strategy in the BitTorrent protocol is realized by combining several peer selection strategies and the rarest-first piece selection strategy, which are summarized in section 3. Legout et al. focus on peer selection strategies [8]. Through several experiments, they observe that those strategies eventually yield clusters of peers with nearly the same upload capacity and within each cluster, pieces are exchanged frequently, and some peers with small upload capacity suffer extremely long file retrieving times. [7] shows that the rarest-first piece selection strategy contributes to achieving the stable system performance, through several simulation experiments.

The fundamental characteristics of peer and piece selection strategies are also studied based on mathematical analyses. Piatek et al. [4] show that the optimistic unchoking (cf. section 3) deteriorates fairness among peers, i.e., the ratio of the download speed to the upload speed tends to be small for peers with large upload capacity. Meulpolder et al. [8] analyze the dynamics generated by the peer selection strategies, using a fluid model. They find that the cluster structure found is not complete and peers with large upload capacity partly provide pieces for those with small upload capacity.

3. Tit-for-Tat-based P2P file distribution

This paper considers a P2P file distribution system like BitTorrent. In what follows, we explain a part of the BitTorrent protocol, which is related closely to the TFT strategy. Every file is divided into small fragments called *pieces*. A new peer tries to retrieve pieces not only from a server but also from other peers. After obtaining a piece, the peer can also transfer it to other peers. In order to encourage peers to exchange pieces with each other, the BitTorrent protocol adopts the rarest-first strategy, where each peer preferentially downloads the rarest piece in the system. Peers who do not finish downloading of the whole file are called *leechers*, while those who finish downloading it are called *seeders*.

Leechers try to retrieve pieces by sending requests to participants of the system, i.e., the server, seeders, and leechers. On receiving requests, the server and seeders altruistically provide the leechers with the requested pieces. Note here that they may require selecting some of the requesting leechers to which they send the pieces, due to the constraint of upload capacity of their own. In the BitTorrent protocol, accepting (resp. declining) requests is called *unchoking* (resp. *choking*). In the current version of the BitTorrent protocol, the sever and seeders adopt an unchoking strategy, where they allocate part of their upload capacity to leechers with high expected download speed and the remaining to those chosen randomly. This is a strategy taking account of the balance between efficiency and fairness.

On the other hand, leechers selfishly select peers to which they upload their owned pieces, according to the following unchoking strategies: Regular unchoking, optimistic unchoking, and anti-snubbing. In the regular unchoking, each leecher unchokes K other leechers who have recently reciprocated with the highest upload speed, where K is a predefined number. The optimistic unchoking is conducted every time interval of a fixed length, e.g., 30 seconds, at which among leechers being unchoked, the leecher with the least reciprocation of the upload speed is choked. Instead, a leecher is chosen randomly among from others and it is unchoked. Finally, in the anti-snubbing, a leecher chokes an unchoked leecher when it cannot download any pieces from it during a time interval of a fixed length, e.g., 60 seconds.

These strategies will eventually balance the numbers of sending and receiving pieces between leechers, and in the framework of game theory, this can be regarded as the TFT strategy. Note, however, that it has not been studied sufficiently on the optimality of such TFT-based P2P file distribution. In this paper, we assume that the TFT strategy is equivalent to maintaining a balance of the numbers of sending and receiving pieces between leechers, and analyze the optimal scheduling in TFT-based P2P file distribution.

4. Modeling TFT-based P2P file distribution and formulation of its optimal scheduling

In this section, we develop a discrete-time model of TFT-based P2P file distribution and formulate its optimal scheduling as a two-step ILP.

4.1 Model

We model TFT-based P2P file distribution in discrete-time. In the system, there are N_{D} *servers*, labeled 1 to N_{D}, and N_{P} *peers*, labeled $N_{\mathrm{D}}+1$ to $N_{\mathrm{D}}+N_{\mathrm{P}}$. Let

$\mathcal{N}_\mathrm{D} = \{1, 2, \ldots, N_\mathrm{D}\}$ and $\mathcal{N}_\mathrm{P} = \{N_\mathrm{D}+1, \ldots, N_\mathrm{D}+N_\mathrm{P}\}$ denote the set of server indices and that of peer indices, respectively. We define $\mathcal{N} = \mathcal{N}_\mathrm{D} \cup \mathcal{N}_\mathrm{P}$ and $N = N_\mathrm{D} + N_\mathrm{P}$. In what follows, the servers and peers are collectively called *nodes*. Note that peers are classified into leechers and seeders, depending on whether file retrieving is finished or not. We assume that all peers try to download a specific file that is divided into M pieces labeled 1 to M. Let $\mathcal{M} = \{1, 2, \ldots, M\}$ denote the set of piece indices.

Let C_i $(i = 1, 2, \ldots, N)$ denote the upload capacity of node i, where $C_i < \infty$. For simplicity, we assume that C_i $(i = 1, 2, \ldots, N)$ is a natural number and the maximum number of pieces that node i can transfer at each time. On the other hand, we assume that the download capacity of each peer is not limited. This assumption is based on the asymmetry of uplink and downlink channel speeds, e.g., ADSL and cable Internet. Note that if the TFT strategy is adopted, peer i's download speed is bound by the total upload capacity of servers and seeders. We model piece transfers between nodes in discrete time. Specifically, piece transfers between nodes are assumed to be synchronized and to be completed in a unit time. We define decision variables $x_{i,j,k}(t)$ $(i, j \in \mathcal{N},\ k \in \mathcal{M},\ t = 1, 2, \ldots, T)$ as

$$x_{i,j,k}(t) = \begin{cases} 1, & \text{if node } i \text{ sends node } j \text{ piece } k \\ & \hfill \text{at time } t, \\ 0, & \text{otherwise,} \end{cases}$$

where T denotes the maximum time that ensures the file retrieval of all peers. In section 4.2, we will show how to determine T. For simplicity in description, let $\mathcal{T}$ and $\mathcal{T}^+$ denote

$$\mathcal{T} = \{0, 1, \ldots, T\}, \quad \mathcal{T}^+ = \{1, 2, \ldots, T\},$$

respectively.

Based on the capacity assumption, each node i can transfer at most C_i pieces at each time. Each node can transfer multiple, different pieces to a specific leecher, under the constraint of the upload capacity. When all peers retrieve all pieces, the file distribution finishes.

We define $z_{i,k}(t)$ $(i \in \mathcal{N},\ k \in \mathcal{M},\ t \in \mathcal{T})$ and $y_i(t)$ $(i \in \mathcal{N},\ t \in \mathcal{T})$ as

$$z_{i,k}(t) = \begin{cases} z_{i,k}(0) + \displaystyle\sum_{s=1}^{t} \sum_{j \in \mathcal{N}} x_{j,i,k}(s), & \text{if } i \in \mathcal{N}_\mathrm{P}, \\ 1, & \text{otherwise,} \end{cases} \tag{1}$$

$$y_i(t) = \begin{cases} 1 - \displaystyle\prod_{k \in \mathcal{M}} z_{i,k}(t) & \text{if } i \in \mathcal{N}_\mathrm{P}, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

respectively, where empty sum is defined to be zero. Note that $z_{i,k}(t)$ represents the state of piece possession of node i at time t, i.e., $z_{i,k}(t) = 1$ if peer i has piece k at time t, and otherwise $z_{i,k}(t) = 0$. On the other hand, $y_i(t)$ represents the role of peer i at time t, i.e., $y_i(t) = 1$ if peer i is a leecher at time t, and otherwise $y_i(t) = 0$, which indicates that peer i is a seeder. Note that servers $(i \in \mathcal{N}_\mathrm{D})$ constantly have all pieces $(z_{i,k}(t) = 1, y_i(t) = 0)$. Therefore we can keep track of the process of the file distribution through $x_{j,i,k}(t)$. Table 1 summarizes notations we introduced.

Table 1 Notations in the model.

Notation	Definition
$\mathcal{N}_\mathrm{D}$	The set of servers, $\{1, 2, \ldots, N_\mathrm{D}\}$
$\mathcal{N}_\mathrm{P}$	The set of peers, $\{N_\mathrm{D}+1, \ldots, N_\mathrm{D}+N_\mathrm{P}\}$
$\mathcal{N}$	The set of nodes, $\{1, 2, \ldots, N\}$ $N = N_\mathrm{D} + N_\mathrm{P},\ \mathcal{N} = \mathcal{N}_\mathrm{D} \cup \mathcal{N}_\mathrm{P}$
$\mathcal{M}$	The set of pieces, $\{1, 2, \cdots, M\}$
C_i	Upload capacity of node i
$x_{i,j,k}(t)$	Decision variables of piece transfers
$y_i(t)$	Decision variables of peers' roles (1: leechers, 0: seeders)
$z_{i,k}(t)$	Decision variables of piece possession (1: possession, 0: missing)

4.2 First Step: Minimization of the average file retrieving time

From a viewpoint of the entire system (i.e., social optimum), an optimal scheduling in P2P file distribution can be regarded as a scheduling that minimizes the average file retrieving time among peers. We thus formulate a minimization problem P_1 of the average file retrieving time as follows.

$$\min \quad \frac{1}{N_\mathrm{P}} \sum_{i \in \mathcal{N}_\mathrm{P}} \tau_i, \tag{3}$$

$$\text{s.t.} \quad z_{i,k}(0) = 1, \quad \forall i \in \mathcal{N}_\mathrm{D}, \forall k \in \mathcal{M}, \tag{4}$$

$$z_{i,k}(0) = 0, \quad \forall i \in \mathcal{N}_\mathrm{P}, \forall k \in \mathcal{M}, \tag{5}$$

$$x_{i,i,k}(t) = 0, \quad \forall i \in \mathcal{N}, \forall k \in \mathcal{M}, t \in \mathcal{T}^+, \tag{6}$$

$$x_{i,j,k}(t) \in \{0, 1\}, \quad \forall i, j \in \mathcal{N}, i \neq j, \forall k \in \mathcal{M}, t \in \mathcal{T}^+, \tag{7}$$

$$x_{i,j,k}(t) \leq z_{i,k}(t-1), \quad \forall i, j \in \mathcal{N},\ i \neq j, \forall k \in \mathcal{M}, t \in \mathcal{T}^+, \tag{8}$$

$$x_{i,j,k}(t) \leq 1 - z_{j,k}(t-1), \quad \forall i, j \in \mathcal{N},\ i \neq j, \forall k \in \mathcal{M}, t \in \mathcal{T}^+, \tag{9}$$

$$\sum_{j \in \mathcal{N}} \sum_{k \in \mathcal{M}} x_{i,j,k}(t) \leq C_i, \quad \forall i \in \mathcal{N}, t \in \mathcal{T}^+, \tag{10}$$

$$\sum_{j \in \mathcal{N}} x_{j,i,k}(t) \leq 1, \quad \forall i \in \mathcal{N}, \forall k \in \mathcal{M}, t \in \mathcal{T}^+, \tag{11}$$

$$\sum_{i \in \mathcal{N}_\mathrm{P}} y_i(T) = 0, \tag{12}$$

$$\sum_{k \in \mathcal{M}} x_{i,j,k}(t) - \sum_{k \in \mathcal{M}} x_{j,i,k}(t) \leq M\left(1 - y_i(t-1) y_j(t-1)\right),$$

$$\forall i, j \in \mathcal{N}, t \in \mathcal{T}^+, \tag{13}$$

$$\tau_i \le \tau_j, \quad \forall i, j \in \mathcal{N}_\mathrm{P},\ C_i \ge C_j. \tag{14}$$

Note that τ_i $(i \in \mathcal{N}_\mathrm{P})$ in (3) denotes peer i's file retrieving time, which is equal to the length of a time interval during which peer i is a leecher:

$$\tau_i = \sum_{t=0}^{T} y_i(t). \tag{15}$$

Note also that T should not be less than $\max_{i\in\mathcal{N}_\mathrm{P}} \tau_i$. We may set T to be, for example,

$$T = \left\lceil M N_\mathrm{P} \left(\sum_{i\in\mathcal{N}_\mathrm{D}} C_i \right)^{-1} \right\rceil,$$

where the right hand side denotes the time required for direct distribution from the servers to all peers.

(4) through (12) represent basic constraints in P2P file distribution. (4) and (5) represent that each server has all pieces at time zero and that each peer has no pieces at time zero, respectively. (6) prohibits nodes from sending any pieces to themselves at each time. (7) permits every node to transfer pieces to others at each time, and (8) is a constraint that at time t, each node can transfer only pieces which the node has at time $t-1$. Furthermore, by (9), transferred pieces is limited to ones which the receiver node do not have. Note here that (7) and (9) ensure that receiver nodes are always leechers. (10) represents the upload capacity constraint. (11) implies that every node can receive a specific piece from at most one node. (12) ensures that all peers will finish downloading the file (i.e., become seeders), by time T.

(13) represents the TFT strategy, under which the numbers of pieces that any pair of leechers exchange should be equal. Note that the left hand side of (13) denotes the difference of the numbers of pieces exchanged between node i and node j at time t, and therefore it ranges $[-M, M]$. When both node i and node j are leechers (i.e., $y_i(t-1) = y_j(t-1) = 1$), the right hand side of (13) is equal to 0, so that the TFT strategy is applied to the piece exchange between leechers. On the other hand, if node i or node j is a server or a seeder (i.e., $y_i(t-1)y_j(t-1) = 0$), this constraint is always holds because the right hand side is equal to M.

(14) defines the order in which peers finish retrieving the file. In this paper, we assume that the file retrieving should be finished in descending order of peers' upload capacity. Note that there are many possibilities in this regard. For example, the system may prioritize accounting peers if they exist.

Note that $y_i(t)$ in (2) and (13) are nonlinear, but it can be linearized, as shown in Appendix A. As a result, problem P_1 can be formulated as an ILP.

4.3 Second step: Fair file retrieving

In general, the optimal solution of problem P_1 in section 4.2 is not unique. Therefore there is room to reduce peers' dissatisfaction with their file retrieving times. Therefore we formulate another problem P_2 in the following manner.

We assume that an ideal scheduling for peer i $(i \in \mathcal{N}_\mathrm{P})$ minimizes his/her file retrieving time under the constraint that all peers finish retrieving the file in descending order of their upload capacity. We define τ'_i $(i \in \mathcal{N}_\mathrm{P})$ as such an ideal file retrieving time of peer i. Note that we can obtain τ'_i for a specific i $(i \in \mathcal{N}_\mathrm{P})$ by solving a modified problem P'_1, where only the objective function is replaced with $\min \tau_i$.

We now define $u_i = \tau_i - \tau'_i$ $(i \in \mathcal{N}_\mathrm{P})$ as a measure of peer i's dissatisfaction. To achieve fairness among all peers, we consider the minimization of the maximum u_i among all peers. Specifically, we formulate problem P_2 by slightly changing P_1, i.e., the objective function is replaced with

$$\min \max_{i\in\mathcal{N}_\mathrm{P}} u_i,$$

and the following constraint is added.

$$\frac{1}{N_\mathrm{P}} \sum_{i\in\mathcal{N}_\mathrm{P}} \tau_i = \tau^*,$$

where τ^* denotes the minimum average file retrieving time obtained by solving problem P_1.

4.4 Peers' behavior after file retrieving

In sections 4.2 and 4.3, we assume that after finishing the file retrieval, peers stay in the system and serve as seeders. We call this a *seeder sojourn scenario* hereafter. When leechers finish retrieving files, however, they may leave the system. In fact, it was pointed out that in actual systems, many seeders leave the system after a relatively short stay in the system and only a limited number of peers altruistically serve as seeders for a long time [10].

In order to examine the influence of peers' behavior after file retrieval, we consider another extreme scenario that all peers leave the system immediately after becoming seeders, which we call a *seeder departure scenario.* To obtain an optimal scheduling in the seeder departure scenario, we have to add the following constraint to P_1 and P_2.

$$x_{i,j,k}(t) \le y_i(t-1), \quad i \in \mathcal{N}_\mathrm{P}, j \in \mathcal{N}, k \in \mathcal{M}, t \in \mathcal{T}^+,$$

which ensures that once peers becomes seeders, they do not provide any pieces for others.

5. Numerical results of the optimal scheduling

In this section, we show some numerical results of the

Table 2 The optimal file retrieving time.

seeders	TFT	A	B	C	D	E	average
sojourn	with	10	11	11	12	12	11.2
sojourn	w/o	11	11	11	11	11	11.0
departure	with	11	11	11	12	12	11.4
departure	w/o	11	11	11	11	11	11.0

Table 3 Comparison of the average file retrieving times in the optimal scheduling and random scheduling.

seeders	TFT	optimal	random
sojourn	with	11.2	13.7
sojourn	w/o	11.0	11.7
departure	with	11.4	14.1
departure	w/o	11.0	11.8

optimal scheduling, which are obtained by solving the two-step ILP with an existing solver CPLEX [12]. We assume a kind of flash crowds [13], where leechers with no pieces start retrieving files from a server at the same time. Such a flash-crowd scenario is the severest situation when peers retrieve the file. Because our objective is to analyze the optimal scheduling and to find the mechanism yielding it, we consider a relatively small system with $N_{\mathrm{D}} = 1, N_{\mathrm{P}} = 5$, and $M = 20$, i.e., one server S tries to distribute twenty pieces to five peers A, B, C, D, and E. We assume that the upload capacity of the peers is heterogeneous, i.e., $C_{\mathrm{S}} = 2$, $C_{\mathrm{A}} = C_{\mathrm{B}} = 3$, $C_{\mathrm{C}} = 2$, and $C_{\mathrm{D}} = C_{\mathrm{E}} = 1$. Note that the average upload capacity of peers is equal to that of the server. In this scenario, the upload capacity of the server is bottleneck because it is not enough to provide a piece for every peer at every time. We use this scenario, unless otherwise stated.

In the above setting, we consider the seeder sojourn and seeder departure scenarios. Note here that the TFT-based P2P file distribution may be inferior to the file distribution without the TFT strategy, because the TFT strategy imposes an additional constraint (13) on piece exchanges. Therefore we also consider the corresponding systems without the TFT strategy. Table 2 presents the optimal file retrieving times of all peers and their average τ^*. We observe that the performance degradation caused by the TFT strategy is small, regardless of the seeders' behavior. We will discuss the reason for this phenomenon in section 6.2.

Next, we consider the effectiveness of the optimal scheduling. For this purpose, we randomly explore a feasible solution of problem P_1, which is called random scheduling. Table 3 compares the average file retrieving time of the optimal scheduling with that of the random scheduling, where the latter is the average of 100 independent simulation runs. We observe that the optimal scheduling becomes more effective when the TFT strategy is adopted.

Finally, we briefly discuss peers' dissatisfaction under the optimal scheduling. Table 4 shows peer i's ($i \in \mathcal{N}_{\mathrm{P}}$) dissatisfaction u_i and their average in the optimal scheduling with the TFT strategy. We observe that the social optimum slightly increases u_i of some peers, especially in the seeder departure scenario.

Table 4 Dissatisfaction with file retrieving time.

seeders	A	B	C	D	E	average
sojourn	0	1	0	1	1	0.6
departure	1	1	0	1	1	0.8

6. Analysis of mechanism yielding optimal scheduling

In this section, we discuss the mechanism that yields the optimal scheduling by analyzing the optimal solution.

6.1 Fundamental features

We first summarize fundamental features to be required for minimizing the average file retrieving time. For simplicity in explanation, we assume that there is one server S and the number M of pieces is a multiple of the upload capacity C_{S} of server S. Note that the following discussion is also applicable to the case of multiple servers by assuming that there is one virtual server whose upload capacity is equal to the total upload capacity of those servers.

It is clear that the lower bound of the file retrieving time of each leecher is given by M/C_{S}, which does not depend on the upload capacity of leechers. Therefore the lower bound τ_{lower} of the average file retrieving time is also given by

$$\tau_{\mathrm{lower}} = M/C_{\mathrm{S}}.$$

On the other hand, the average file retrieving time is bounded above when server S directly distributes M pieces to every leechers. In order to minimize the average file retrieving time in the direct distribution, server S has to complete file distribution to all peers one by one. In this case, i-th peer's file retrieval time is $i(M/C_{\mathrm{S}})$, so that the upper bound τ_{upper} is given by

$$\tau_{\mathrm{upper}} = N_{\mathrm{P}}^{-1} \sum_{i \in \mathcal{N}_{\mathrm{P}}} i(M/C_{\mathrm{S}}) = \frac{M(N_{\mathrm{P}}+1)}{2C_{\mathrm{S}}}.$$

Because this worst average file retrieving time increases with the number N_{P} of peers, it is essential to encourage peers to exchange pieces each other. Note here that piece exchanges have the following fundamental features.

(i) Nodes can upload pieces only to leechers without those pieces, regardless of the TFT strategy. The TFT strategy further requires this constraint in both directions between two leechers.

(ii) Under the TFT strategy, leecher i with $C_i > C_{\mathrm{S}}$ has a good chance of achieving the average download speed C_{S} over his/her file retrieval even if leecher i temporarily suffers from slower download speed than C_{S}. Note that if this is the case, the file retrieving time is minimized, i.e., $\tau_i = \tau_{\text{lower}}$.

(iii) Under the TFT constraint, leechers i with $C_i \leq C_{\text{S}}$ requires more piece transfers from server S than those with $C_i > C_{\text{S}}$ because they can download at most C_i pieces from other leechers at each time, in order to achieve the minimum file retrieving time τ_{lower}.

In what follows, we examine the desirable strategy for piece transfers in consideration of the above fundamental features. As far as the behavior of seeders is concerned, we mainly consider the seeder departure scenario because the server plays the leading role in piece transfers in this scenario and as stated, many seeders leave the system after a relatively short stay in actual systems [10].

6.2 Optimal strategy in the seeder departure scenario

To satisfy feature (i) between as many leechers as possible, the server has to provide rare piece(s) for each of them, which is nothing but the rarest-first strategy adopted in the BitTorrent protocol. Note here that rare pieces that the server provides should be different for different leechers so as to accelerate piece exchanges.

Considering features (ii) and (iii) in section 6.1, as well as feature (i) of the rarest-first strategy, we can deduce a desirable piece distribution process that shortens the average file retrieving time under the TFT strategy. We define $\mathcal{L}_{\text{L}}$ and $\mathcal{L}_{\text{H}}$ as

$$\mathcal{L}_{\text{L}} = \{l;\ l \in \mathcal{N}_{\text{P}}, C_l \leq C_s\},\ \mathcal{L}_{\text{H}} = \{l;\ l \in \mathcal{N}_{\text{P}}, C_l > C_s\},$$

respectively.

(a) At an early stage of the file distribution, server S sends pieces preferentially to leechers in $\mathcal{L}_{\text{L}}$. Those pieces should be different so as to accelerate piece exchanges between leechers. Meanwhile, leechers in $\mathcal{L}_{\text{H}}$ have no pieces.

(b) When leechers in $\mathcal{L}_{\text{L}}$ have enough pieces through step (a), server S sends new pieces to leechers in $\mathcal{L}_{\text{H}}$, and then those leechers exchange their pieces with those in $\mathcal{L}_{\text{L}}$. As a result, they can rapidly retrieve pieces that the server distributed over the system.

In what follows, we confirm the above process (a) and (b) through the numerical results of the optimal scheduling, where system parameters are identical to those in section 5 unless otherwise stated, and leechers and peers are used interchangeably.

Fig. 1 shows an optimal file retrieval process in the seeder departure scenario with the TFT strategy. We give the time step in x-axis and the amount of pieces that each peer i retrieves by t, i.e., $\sum_{k \in \mathcal{M}} z_{i,k}(t)$, in y-axis. From Fig. 1, we observe that leechers D and E in $\mathcal{L}_{\text{L}}$ have more pieces than others at the initial stage and that leechers A and B in $\mathcal{L}_{\text{H}}$ start acquiring

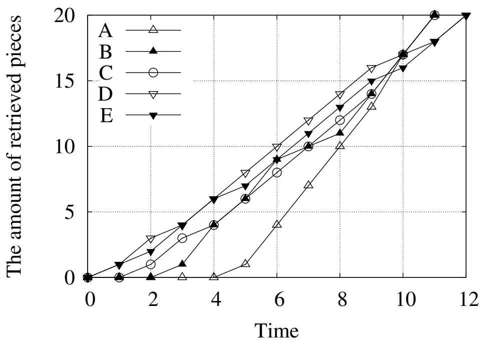

Fig. 1 Optimal file retrieval process (seeder departure scenario with TFT).

Table 5 The number of pieces retrieved from server S (seeder departure scenario with TFT).

A	B	C	D	E
2	1	3	9	9

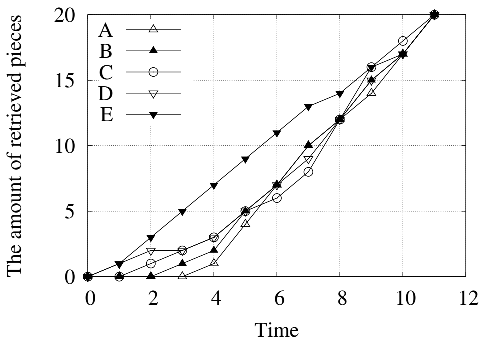

Fig. 2 Optimal file retrieval process (seeder departure scenario with TFT, $C_{\text{S}} = 2$, $C_{\text{A}} = C_{\text{B}} = C_{\text{C}} = C_{\text{D}} = 3$, and $C_{\text{E}} = 1$).

Table 6 The number of pieces retrieved from server S (seeder departure scenario with TFT, $C_{\text{S}} = 2$, $C_{\text{A}} = C_{\text{B}} = C_{\text{C}} = C_{\text{D}} = 3$, and $C_{\text{E}} = 1$)).

A	B	C	D	E
3	3	3	3	10

pieces later, at a higher average download speed than $C_{\text{S}} = 2$. Table 5 shows the number of pieces that each peer retrieves from server S. Peers D and E in $\mathcal{L}_{\text{L}}$ receive more pieces from server S than peers A and B in $\mathcal{L}_{\text{H}}$. These results agrees with the above-mentioned process (a) and (b), i.e., it is crucial to distribute pieces from server S indirectly to leechers in $\mathcal{L}_{\text{H}}$ via those in $\mathcal{L}_{\text{L}}$.

Next, we focus on the ratio $|\mathcal{L}_{\text{H}}|/|\mathcal{L}_{\text{L}}|$ of the number of peers in $\mathcal{L}_{\text{H}}$ to that in $\mathcal{L}_{\text{L}}$. Because the default scenario satisfies $|\mathcal{L}_{\text{H}}|/|\mathcal{L}_{\text{L}}| < 1$, we further examine the case of $|\mathcal{L}_{\text{H}}|/|\mathcal{L}_{\text{L}}| > 1$. Fig. 2 shows an optimal file retrieval process in the seeder departure scenario with the TFT strategy and the capacity distribution: $C_{\text{S}} = 2$, $C_{\text{A}} = C_{\text{B}} = C_{\text{C}} = C_{\text{D}} = 3$, and $C_{\text{E}} = 1$. Table 6 shows the number of pieces that each peer retrieves from server S in that scenario. Fig. 2 and Table 6 show that the above-mentioned process (a) and (b) is also realized in this scenario.

Fig. 3 shows an optimal file retrieval process in the

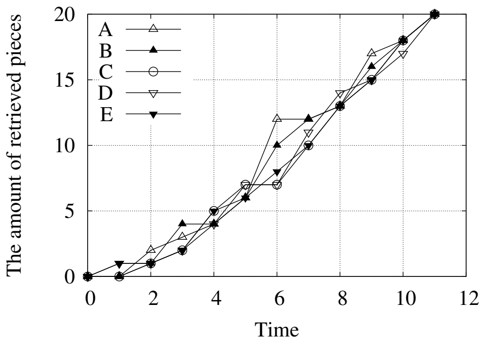

Fig. 3 Optimal file retrieval process (seeder departure scenario without TFT).

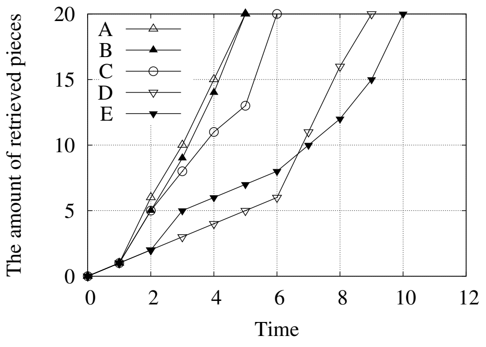

Fig. 4 Optimal file retrieval process (seeder departure scenario with TFT), $C_S = 5$).

seeder departure scenario without the TFT strategy. We do not observe any significant differences among file retrieval processes of five peers, regardless of their upload capacity. The reason is that if the TFT strategy is not adopted, any peers do not have to follow the above-mentioned process (a) and (b). Recall that the degradation of the average file retrieving time, which is caused by the TFT strategy, is small as shown in Table 2. Comparing Fig. 1 with Fig. 3, we observe that the optimal scheduling with the TFT can recover the average upload speed at the latter stage of the file distribution, with the help of process (a) and (b).

Finally, we show a numerical example when the upload capacity C_S of server S is large enough to send a piece to every peer at every time. Fig. 4 illustrates an optimal file retrieval process, where $C_S = 5$ and all other parameters are identical to those in section 5. Note that all leechers are in $\mathcal{L}_L$ in this setting. Peers are distinguished clearly according to their upload capacity. In the optimal scheduling of this scenario, each leecher retrieves one piece from server S at time 1, and at time 2, piece exchanges are performed between pairs of leechers (A,B), (A,C), (A,D), (B,C), and (B,E), and server S sends new pieces only to leechers A, B, and C. Note that upload capacity of leechers is utilized fully at time 2.

In actual systems, the upload capacity of the server tends to be bottleneck due to the increase of the number

Table 7 The number of pieces retrieved from server S (with TFT).

	scenario	A	B	C	D	E
$C_S = 2$	departure	2	1	3	9	9
	sojourn	6	4	3	7	4
$C_S = 5$	departure	8	8	10	12	12
	sojourn	11	8	6	8	7

of peers. Therefore, the case of $C_S = 2$ is more important than that of $C_S = 5$. Recall that the conventional piece transfer strategy of servers/seeders makes much account of the download speed of leechers and fairness among them (see section 3). Even though it is intuitively acceptable, but our results indicate that piece flows from leechers with small upload capacity to those with large capacity play an important role in minimizing the average file retrieving time.

6.3 Seeder sojourn scenario: Consideration of peers' altruism

So far, we have examined only the seeder departure scenario. In this section, we briefly discuss the seeder sojourn scenario. Fig. 5 illustrates an optimal file retrieval process in the seeder sojourn scenario with the TFT strategy. We observe that the piece retrieval process at the early stage of the file distribution is similar to that in Fig. 1, which indicates that process (a) and (b) are conducted. On the other hand, we also observe that peers in $\mathcal{L}_H$ are preferentially treated at the latter stage of the file distribution, compared with the seeder departure scenario.

In this scenario, once a peer becomes a seeder, it fills the role of servers. As stated, a group of the server and seeders is regarded as a virtual server whose upload capacity is given by the sum of their capacity. In our formulation, leechers in $\mathcal{L}_H$ becomes seeders sooner than those in $\mathcal{L}_L$ do (cf. (14)), and leechers in $\mathcal{L}_L$ can retrieve files not only from server S but also seeders. By doing so, the average file retrieving time is shortened.

We can recognize this phenomenon typically by seeing the number of pieces retrieved directly from server S, as shown in Table 7. Peers with large upload capacity in the seeder sojourn scenario retrieve more pieces from server S than those in the seeder departure scenario. It should be noted that the preferential treatment of altruistic peers leads the system to its social optimum.

7. Conclusion

In this paper, we analyzed the optimal scheduling in TFT-based P2P file distribution. We first formulated the scheduling in the TFT-based P2P file distribution as two-step ILP. We obtained the optimal solution by an existing solver in several scenarios. From numerical results of the optimal scheduling, we observed that the TFT strategy causes only a small increase in the

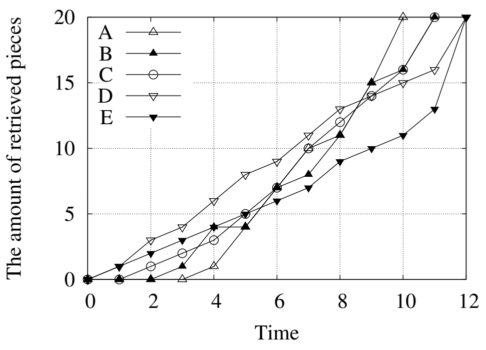

Fig. 5 Optimal file retrieval process (seeder sojourn scenario with TFT).

average file retrieval time.

An analysis of the optimal solutions revealed that the following mechanism yielded the optimal scheduling. First, the rarest-first strategy is effective, which agrees with the past studies. In order to minimize the average file retrieving time, however, the relationship in size between the upload capacity of peers and of the server is important; pieces should be distributed from the server indirectly to peers with large upload capacity via those with small upload capacity.

Finally, we should note that the optimal scheduling can be achieved mainly by the above-mentioned server's piece transfer strategy. On the contrary, peers only make their best effort to retrieve their demanded pieces from others, under the TFT constraint. In other words, this paper opens up a new vista of the controllable P2P file distribution systems where the server can almost control the state of piece possession of peers with the help of the TFT strategy and the above-mentioned piece transfer strategy based on peers' information, i.e., the state of piece possession, the upload capacity, and altruism.

Appendix A: Linearization of products of binary variables

The product of variables are nonlinear but it can be transformed into linear expressions if all variables are binary [11]. In particular,

$$y = x_1 x_2 \cdots x_k, \qquad x_i = \{0,1\}, \quad (i = 1,2,\ldots,k)$$

can be rewritten to be the following linear expressions:

$$(k-1) - \sum_{i=1}^{k} x_i + y \geq 0,$$
$$x_i - y \geq 0, \quad x_i = \{0,1\}, \quad (i = 1,2,\ldots,k).$$

With this technique, nonlinear terms in (2) and (13) can be linearized.

References

[1] "BitTorrent," `http://www.bittorrent.com/`.

[2] D. Hughes, G. Coulson and J. Walkerdine: "Free Riding on Gnutella Revisited: The Bell Tolls?," IEEE Distributed Systems Online, **6**, 6, pp. 1–18 (2005).

[3] S. Jun and M. Ahamad: "Incentives in BitTorrent Induce Free Riding," Proc. of the 2005 ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems, pp. 116–121 (2005).

[4] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy and A. Venkataramani: "Do Incentives Build Robustness in BitTorrent," Proc. of the 4th USENIX Symposium on Networked Systems Design & Implementation, pp. 1–14 (2007).

[5] M. Meulpolder, J. Pouwelse, D. Epema and H. Sips: "Modeling and Analysis of Bandwidth-Inhomogeneous Swarms in BitTorrent," Proc. of the 9th International Conference on Peer-to-Peer Computing 2009 (P2P'09), pp. 232–241 (2009).

[6] K. Eger and U. Killat: "Bandwidth Trading in BitTorrent-like P2P Networks for Content Distribution," Computer Communications, **31**, 2, pp. 201–211 (2008).

[7] P. Felber and E. W. Biersack: "Self-Scaling Networks for Content Distribution," Proc. of International Workshop on Self-* Properties in Complex Information Systems, pp. 1–14 (2004).

[8] A. Legout, N. Liogkas, E. Kohler and L. Zhang: "Clustering and Sharing Incentives in BitTorrent Systems," ACM SIGMETRICS Performance Evaluation Review, **35**, 1, pp. 301–312 (2007).

[9] R. L. Xia and J. K. Muppala: "A Survey of BitTorrent Performance," IEEE Communications Surveys & Tutorials, **12**, 2, pp. 140–158 (2010).

[10] J. Pouwelse, P. Garbacki, D. Epema and H. Sips: "The Bittorrent P2P File-Sharing System: Measurements and Analysis," Peer-to-Peer Systems IV, pp. 205–216 (2005).

[11] D. Chen, R. G. Batson and Y. Dang: "Applied Integer Programming," Wiley (2010).

[12] "IBM ILOG CPLEX Optimizer," `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

[13] X. Yang and G. de Veciana: "Service Capacity of Peer to Peer Networks," Proc. of IEEE INFOCOM 2004, pp. 2242–2252 (2004).

Masashi Hasegawa received the B.E. and M.E. degrees in information communication engineering from Osaka University, Japan in 2012 and 2014, respectively. His main research interests include Peer-to-Peer networking. He is currently with Central Nippon Expressway Company Limited, Japan.

Masahiro Sasabe is currently an Associate Professor in the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. He received the M.E. and Ph.D. degrees from Osaka University, Japan, in 2003 and 2006, respectively. From 2003 to 2004, he was a Research Fellow with 21COE-JSPS, Japan. From 2004 to 2007, he was an Assistant Professor in the Cybermedia Center, Osaka University. From 2007 to 2014, he was an Assistant Professor in the Department of Information and Communication Technology, Graduate School of Engineering, Osaka University. His research interests include P2P/overlay networking, DTNs, and game-theoretic approach. Dr. Sasabe is a member of IEEE.

Tetsuya Takine is currently a Professor in the Department of Information and Communications Technology, Graduate School of Engineering, Osaka University, Suita, Japan. He was born in Kyoto, Japan, on November 28, 1961, and received B.Eng., M.Eng., and Dr.Eng. degrees in applied mathematics and physics from Kyoto University, Kyoto, Japan, in 1984, 1986, and 1989, respectively. In April 1989, he joined the Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University, as an Assistant Professor. Beginning in November 1991, he spent one year at the Department of Information and Computer Science, University of California, Irvine, on leave of absence from Kyoto University. In April 1994, he joined the Department of Information Systems Engineering, Faculty of Engineering, Osaka University as a Lecturer, and from December 1994 to March 1998, he was an Associate Professor in the same department. From April 1998 to May 2004, he was an Associate Professor in the Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University. His research interests include queueing theory, emphasizing numerical computation, and its application to performance analysis of computer and communication networks. He is now serving as an associate editor of Queueing Systems, Stochastic Models, International Transactions in Operational Research, and Journal of Operations Research Society of Japan. He received Telecom System Technology Award from TAF in 2003 and 2010, Best Paper Awards from ORSJ in 1997, from IEICE in 2004 and 2009, and from ISCIE in 2006, Best Tutorial and Magazine Awards from IEICE ComSoc in 2012, and three best paper awards at international conferences. Dr. Takine is a fellow of ORSJ and a member of IEEE, IPSJ, and ISCIE.