Analysis of Optimal Piece Flow in Tit-for-Tat-Based P2P Streaming

Masahiro Sasabe*

Graduate School of Science and Technology, Nara Institute of Science and Technology, 8916-5 Takayama-cho, Ikoma, Nara 630-0192, Japan

Abstract

BitTorrent, which is one of the successful Peer-to-Peer (P2P) file distribution systems, adopts the tit-for-tat (TFT) strategy in game theory to encourage cooperation among peers, i.e., each peer has to provide fragments of the original file, called pieces, to others so as to retrieve its demanding pieces from them. Because the TFT strategy can restrict free riding behavior of peers, there are also several TFT-based P2P streaming systems and the performance of such existing systems has been analyzed. However, optimal piece flow in TFT-based P2P streaming has not been revealed yet. In this paper, a discrete-time model of TFT-based P2P streaming is first developed and integer linear programming (ILP) is formulated to determine the optimal piece flow where the average play-out delay is minimized. By solving the ILP using existing solver, i.e., CPLEX, we can obtain numerical examples of optimal piece flow. The analysis of obtained optimal piece flow reveals that (1) optimal piece selection is based on the balance between in-order piece retrieving and the rarest-first piece retrieving, (2) optimal peer selection depends on the upload capacities of peers and the stage of streaming, (3) the number of pieces does not affect the system performance, (4) the maximum play-out delay can be bounded by the ratio of the number of peers to the server's upload capacity, and (5) how the relaxation of TFT constraint can improve the system performance.

Keywords: P2P streaming, optimal piece flow, tit-for-tat strategy, integer linear programming (ILP)

1. Introduction

With the proliferation of the Internet streaming services, e.g., YouTube [1], Hulu [2], and NetFlix [3], IP video traffic will be 82% of all IP traffic (both business and consumer) by 2020, up from 70% in 2015 [4]. Most of the current streaming services adopt client-server architectures by taking account of the simplicity of managing user information and contents. Such client-server architectures, however, have potential drawbacks of load concentration and single point of failure at the server(s).

Peer-to-Peer (P2P) streaming has been expected to overcome such drawbacks of client-server architectures by utilizing the upload capacities of peers joining the streaming services [5]. There are some P2P streaming services, e.g., BitTorrent Live [6], PPTV [7], and GridCast [8], where, in the P2P architectures, contents are divided into fragments called pieces. Clients called peers can retrieve the pieces not only from the server(s) but also from other peers. As a result, the streaming system can dynamically and autonomously increase/decrease the scale depending on the number of peers [9]. Such scalability, however, requires peers' altruistic behavior, i.e., uploading pieces to others. In case of streaming services, users tend to hesitate in uploading pieces to others and become free riders, due to high and long-term bandwidth consumption [10, 11].

To alleviate such free-riding problems, Tit-for-Tat (TFT) strategy in game theory has been expected to encourage peers to willingly exchange pieces with others [9, 10, 12, 13, 14]. Note that BitTorrent [15] first applied the TFT strategy to P2P file distribution systems. The TFT strategy in P2P content distribution means that each peer has to upload pieces to others so as to retrieve his/her demanding pieces from them. To exchange pieces between two peers, they have to possess different pieces. As a result, in case of file distribution, it is rational and optimal for peers to follow a *rarest-first* strategy, where they preferentially retrieve the rarest piece(s) in the system.

In case of streaming distribution, however, such a rarest-first strategy is not necessarily optimal because *in-order* strategy is also important, where pieces are prioritized in play-out order. There are several studies on performance evaluation of TFT-based P2P streaming [16, 17, 18, 19]. However, most of them have been focusing on the piece retrieving strategies that seem to be rational, e.g., the rarest-first strategy and in-order strategy. As a result, the optimal piece flow in TFT-based P2P streaming has not been revealed yet.

In this paper, the optimal piece flow in TFT-based P2P streaming is analyzed with the help of integer linear programming (ILP). Although our problem is NP-hard, it can be solved by existing solver, e.g., CPLEX [20], in case of small-scale systems. The analysis of the obtained optimal solution will show fundamental characteristics of the optimal piece flow.

The main contributions of this paper are as follows:

1. A discrete-time model of TFT-based P2P streaming is developed and ILP is formulated to determine the optimal piece flow where the average play-out delay is minimized.

*Corresponding author
Email address: sasabe@is.naist.jp (Masahiro Sasabe)

By solving the ILP using existing solver, numerical examples of optimal piece flow can be obtained.

2. The analysis of obtained optimal piece flow reveals that (1) optimal piece selection is based on the balance between in-order piece retrieving and the rarest-first piece retrieving, (2) optimal peer selection depends on the upload capacities of peers and the stage of streaming, (3) the number of pieces does not affect the system performance, (4) the maximum play-out delay can be bounded by the ratio of the number of peers to the server's upload capacity, and (5) how the relaxation of TFT constraint can improve the system performance.

The rest of the paper is organized as follows. Section 2 gives related work. The determination of optimal piece flow in TFT-based P2P streaming is formulated as ILP in Section 3. Section 4 demonstrates some numerical results. Finally, Section 5 gives concluding remarks.

2. Related work

Incentive mechanism design plays the important role of alleviating the free-riding problem in P2P content distribution [21, 22, 23, 24]. Since the P2P systems consist of anonymous and potentially selfish users, game theoretic approaches are suitable for dealing with competitive situations among such users: TFT approach [23], Stackelberg game approach [22], and auction-based approach [24]. Most of the existing incentive mechanisms can be classified into two categories: direct reciprocity and indirect reciprocity. In the direct reciprocity, as represented by the TFT strategy, each peer evaluates the contribution of other peers based only on the amount of data that it obtained from the corresponding peer. On the other hand, the indirect reciprocity evaluates the contribution of each peer based on the whole amount of data that the corresponding peer provided to others. In this paper, we focus on the TFT strategy because of its simplicity and robustness against peers' cheating of their contribution.

To improve the system performance of P2P streaming, e.g., play-out delay, piece and peer selection policies have also been studied. Fan et al. investigate how the three kinds of piece selection policies, i.e., rarest random, naive sequential, and cascading, affect the balance between throughput, robustness, and in-order delivery in case of on-demand streaming [25]. In [26], authors propose an abstract stochastic model to analyze and compare the content-diversified oriented policy and importance-first oriented policy in live streaming. Based on the model, they also propose a method that dynamically switches these two policies. In [27], authors propose a sophisticated peer selection strategy under a P2P system where two kinds of services, i.e., streaming service and file downloading service, coexist. In [28], authors propose decentralized peer selection strategies to achieve semi-optimal streaming capacity in large-scale P2P VoD systems with sparse connectivity among peers. In [29], the authors consider a streaming system where a server distributes pieces in play-out order, i.e., in-order policy, and each peer sends the newest piece among the received pieces to others, i.e., newest-first policy. In this system, they mathematically analyze the system performance, e.g., the lower bound of initial play-out delay, with the help of trellis graph techniques. Note that these works do not consider the TFT-based P2P streaming.

There are several studies on performance evaluation of TFT-based P2P streaming: analytical approaches [16, 17, 18, 30] and simulation-based approach [19].

Tewari and Klienrock provide an analytical model to design the BitTorrent-based live video streaming solutions [30]. They show that existing server-based streaming infrastructures with a well-designed peer-to-peer solution can achieve allow higher streaming rates and the efficiency of the solution depends on the group size of peers and the number of pieces available for sharing at any given time.

Parvez et al. develop analytic models to characterize the behavior of BitTorrent-like on-demand stored media content delivery [16]. These models can capture the effects of different piece selection policies, e.g., rarest-first and two variants of in-order. Through the analysis, they show that one of the in-order variants is optimal in terms of play-out delay among these three policies. The authors also extend this study to analyze two additional probabilistic piece selection policies (Portion and Zipf), which try to cope with the trade-off between the rarest-first policy and in-order policy [17].

In [18], authors provide a mathematical model to analyze the hybrid BitTorrent system composed of both downloading peers and streaming peers. They propose a sliding window-based hybrid method that combines the rarest-first policy with the in-order policy to support the two-types of peers. The experimental results show that the proposed method can achieve higher throughput and better streaming continuity than the sequential policy.

In [19], authors propose a simulation-based methodology to provide a common basis to compare the performance of many P2P streaming protocols under a variety of conditions. They show that, despite of their considerable differences, all of the existing BitTorrent-like P2P streaming protocols share some characteristics, i.e., their bandwidth reciprocity based methods to encourage cooperation cannot necessarily optimize the overall performance.

All of the above approaches focus on the performance of piece selection policies that seem to be rational. In this paper, the optimal piece flow in TFT-based P2P streaming is analyzed, which yields the minimum average play-out delay among peers, with the help of ILP formulation. As for TFT-based P2P file distribution, Hasegawa et al. have already studied the optimal piece flow, which achieves the minimum average file retrieving time [31]. Inspired by the approach in [31], ILP formulation for TFT-based P2P streaming is newly developed.

3. Modeling TFT-based P2P streaming and formulation of determination of its optimal piece flow

In case of the streaming service, each user wants to shorten the initial waiting time for play-out, called *play-out delay*. From the viewpoint of overall system, average and maximum play-out delay among users should be minimized. In this section, a discrete-time model of TFT-based P2P streaming is developed and the determination of its optimal piece flow is formulated as three-step ILP. First two steps are used to obtain optimal flows yielding low-latency streaming and the last step is used to find out simple one from the obtained optimal flows.

3.1. Model

TFT-based P2P streaming is modeled as a discrete-time system where there are N_D *servers*, denoted by $\mathcal{N}_\mathrm{D} = \{1, 2, \ldots, N_\mathrm{D}\}$, and N_P *peers*, denoted by $\mathcal{N}_\mathrm{P} = \{N_\mathrm{D}+1, \ldots, N_\mathrm{D}+N_\mathrm{P}\}$. 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 hereinafter referred to as *nodes*. Peers are classified according to the state of file retrieving, e.g., *seeds* that complete file retrieving and *leechers* that are under file retrieving. Suppose that all peers aim to play a specific media divided into M pieces, denoted by $\mathcal{M} = \{1, 2, \ldots, M\}$.

The upload capacity of node i ($i = 1, 2, \ldots, N$) is denoted by C_i, which is assumed to be a natural and finite number. Each node can send at most C_i pieces in a unit time. On the contrary, the download capacity of each peer is assumed to be unlimited by taking account of the fact that the uplink and downlink channel speeds are asymmetry in the Internet, e.g., ADSL and cable Internet. Note that applying the TFT strategy bounds the peer i's download speed by the total upload capacity of servers and seeds. The piece transfers between nodes in discrete time, i.e., *piece flow*, are modeled by decision variables $x_{t,k,i,j}$ ($t = 1, 2, \ldots, T$, $k \in \mathcal{M}$, $i, j \in \mathcal{N}$) as

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

where T denotes the maximum time until that all peers finish file retrieving. In section 3.2, the determination of T will be shown. For simplicity in description, two sets of time steps, $\mathcal{T}$ and $\mathcal{T}^+$, are defined as

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

For simplicity of explanation, in what follows, we formulate the problem under the assumption where all peers join the system at $t = 0$ but we can easily extend the formulation such that each peer joins in an asynchronous manner, by permitting each peer $i \in \mathcal{N}_\mathrm{P}$ to send and receive pieces only after its join time $t_i \geq 0$. In other words, the proposed approach can deal with both live streaming and on-demand streaming. Note that on-demand streaming is more complicated than live streaming because it can support multiple controlling methods (user behavior) in addition to playing, e.g., pausing, skipping, fast-forwarding, and rewinding. In this paper, we only consider playing method but other controlling methods under a specific scenario can also be considered by adding corresponding constraints.

Based on piece flow $x_{t,k,i,j}$ ($t = 1, 2, \ldots, T$, $k \in \mathcal{M}$, $i, j \in \mathcal{N}$), the state of piece k's possession of node i at time t, $z_{t,k,i}$ ($t \in \mathcal{T}$, $k \in \mathcal{M}$, $i \in \mathcal{N}$), and the role of node i at time t, i.e., $y_{t,i}$ ($t \in \mathcal{T}$, $i \in \mathcal{N}$), can also be defined as follows:

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
$x_{t,k,i,j}$	Decision variables of piece transfers
$y_{t,i}$	Variables of nodes' roles (1: leechers, 0: servers/seeds)
$z_{t,k,i}$	Variables of piece possession (1: possession, 0: missing)
w_i	Play-out delay of peer i
τ_i	File retrieving time of peer i
Q	The number of pieces that servers initially have

$$z_{t,k,i} = \begin{cases} z_{0,k,i} + \sum_{s=1}^{t} \sum_{j \in \mathcal{N}} x_{s,k,j,i}, & \text{if } i \in \mathcal{N}_\mathrm{P}, \\ 0, & \text{if } i \in \mathcal{N}_\mathrm{D},\ k > t + Q, \\ 1, & \text{if } i \in \mathcal{N}_\mathrm{D},\ k \leq t + Q, \end{cases} \tag{1}$$

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

In case of peers, $z_{t,k,i} = 1$ if peer i has piece k at time t, and otherwise $z_{t,k,i} = 0$. In case of servers, $z_{t,k,i}$ is set to be 0 or 1, depending on the relationship between k, t, and Q, where Q represents range $[1, Q]$ of piece id that the servers possess at $t = 0$. Q ranges from 1 to M and is determined according to the type of media, i.e., stored media or live media. If $Q = M$, the servers initially store all the pieces, and thus this is a stored-media case. Otherwise, they initially store $Q < M$ pieces, which can be regarded as a live-media case where we assume the servers generate a new piece at each time step and start the streaming service after Q pieces are generated. Note that smaller Q will regulate the diversity of pieces among peers at the initial stage of streaming service. $y_{t,i} = 1$ if peer i is a leecher at time t, and otherwise $y_{t,i} = 0$, which indicates that peer i is a seed at time t. As for the servers, $y_{t,i} = 0$ ($t \in \mathcal{T}, i \in \mathcal{N}_\mathrm{D}$). As a result, the streaming process can be tracked by $x_{t,k,j,i}$. Table 1 summarizes notations used.

3.2. First step: Minimization of the average play-out delay

From the viewpoint of social optimum, an optimal piece flow in P2P streaming can be regarded as a piece flow that minimizes the average play-out delay among peers:

$$\overline{w} = \frac{1}{N_\mathrm{P}} \sum_{i \in \mathcal{N}_\mathrm{P}} w_i.$$

Here w_i ($i \in \mathcal{N}_\mathrm{P}$) denotes peer i's play-out delay,

$$w_i = \max_{k \in \mathcal{M}} w_{i,k}, \quad \forall i \in \mathcal{N}_\mathrm{P}, \tag{3}$$

where $w_{i,k} = \sum_{t=0}^{T} (1 - z_{t,k,i}) - (k-1)$ is the waiting time for playing out piece k. Note that $\sum_{t=0}^{T} (1 - z_{t,k,i})$ represents the time to finish retrieving piece k and $k-1$ is the time at which peer i can play out piece k if it has that piece at $t = 0$. As a result, peer i can smoothly play out all pieces if it accepts the initial play-out delay of w_i. In other words, w_i can also be regarded as the maximum jitter that peer i experiences if it starts streaming without any initial play-out delay.

Note that T should be large enough to finish distributing all pieces to all peers, which can be bounded by T_max:

$$T_\mathrm{max} = \left\lceil M N_\mathrm{P} \left(\sum_{i \in \mathcal{N}_\mathrm{D}} C_i \right)^{-1} \right\rceil, \tag{4}$$

which is the time required for the servers to directly distribute all pieces to all peers.

As a result, the first-step problem P_1 to minimize the average play-out delay is formulated as follows.

$$\begin{aligned}
\min \quad & \overline{w}, & (5)\\
\text{s.t.} \quad & z_{0,k,i} = 1, \quad \forall k \in \mathcal{M}, \forall i \in \mathcal{N}_\mathrm{D}, & (6)\\
& z_{0,k,i} = 0, \quad \forall k \in \mathcal{M}, \forall i \in \mathcal{N}_\mathrm{P}, & (7)\\
& x_{t,k,i,i} = 0, \quad \forall t \in \mathcal{T}^+, \forall k \in \mathcal{M}, \forall i \in \mathcal{N}, & (8)\\
& x_{t,k,i,j} \in \{0, 1\}, \quad \forall t \in \mathcal{T}^+, \forall k \in \mathcal{M}, \forall i, j \in \mathcal{N}, i \neq j, & (9)\\
& x_{t,k,i,j} \le z_{t-1,k,i}, \ \forall t \in \mathcal{T}^+, \forall k \in \mathcal{M}, \forall i, j \in \mathcal{N}, \ i \neq j, & (10)\\
& x_{t,k,i,j} \le 1 - z_{t-1,k,j}, \\
& \qquad \forall t \in \mathcal{T}^+, \forall k \in \mathcal{M}, \forall i, j \in \mathcal{N}, \ i \neq j, & (11)\\
& \sum_{j \in \mathcal{N}} \sum_{k \in \mathcal{M}} x_{t,k,i,j} \le C_i, \quad \forall t \in \mathcal{T}^+, \forall i \in \mathcal{N}, & (12)\\
& \sum_{j \in \mathcal{N}} x_{t,k,j,i} \le 1, \quad \forall t \in \mathcal{T}^+, \forall k \in \mathcal{M}, \forall i \in \mathcal{N}, & (13)\\
& \sum_{i \in \mathcal{N}_\mathrm{P}} y_{T,i} = 0, & (14)\\
& \sum_{k \in \mathcal{M}} (x_{t,k,i,j} - x_{t,k,j,i}) \le M \left(1 - y_{t-1,i} y_{t-1,j}\right), \\
& \qquad \forall t \in \mathcal{T}^+, \forall i, j \in \mathcal{N}, & (15)\\
& w_i \le w_j, \quad \forall i, j \in \mathcal{N}_\mathrm{P}, \ i < j, \ C_i \ge C_j. & (16)
\end{aligned}$$

Since both P2P streaming and P2P file distribution share the nature of P2P systems, (6) through (15) become the same constraints in P2P file distribution [31]. The basic constraints in P2P streaming are given by (6) through (14). (6) and (7) represent that each server has all pieces at $t = 0$ while each peer has no piece at $t = 0$, respectively. (8) forbids nodes to send any pieces to themselves at time t. (9) allows nodes to send pieces to others at time t. (10) indicates that at time t each node can transfer only pieces which the node has at time $t-1$. (11) limits transferred pieces to ones which the receiver node do not have. Note that (9) and (11) ensure that only leechers can retrieve pieces. The upload capacity constraint is given by (12). (13) ensures that every node receives a piece from at most one node. (14) ensures that all peers will finish file retrieving, i.e., become seeds, until $t = T$.

(15) gives the TFT strategy where any pair of leechers should exchange the same number of pieces. The left hand side of (15) is the difference of the numbers of piece exchanged between node i and node j at time t. Simply, this should be zero but the right hand side of (15) is generalized to support the case when at least one of nodes i and j is a server or seed. In such a case, the right hand side becomes the number of pieces, M, and thus (15) is always satisfied.

(16) ensures that the play-out delay should be in descending order of peers' upload capacity by taking account of the fact that peers with higher upload capacity can contribute to faster piece distribution. This constraint can also be replaced according to the policy of the server, e.g., prioritization of accounting peers.

Finally, $y_{t,i}$ in (2) and (15) are nonlinear but they can be linearized (See Section A). Consequently, problem P_1 can be formulated as ILP.

3.3. *Second step: Minimization of the maximum play-out delay among peers*

By solving problem P_1, we may obtain more than one optimal solution, all of which minimize the average play-out delay among peers. In such cases, there may be a chance to refine the optimal piece flow in terms of reducing the maximum play-out delay among peers while keeping the minimized average play-out delay. The maximum play-out delay can be defined as

$$w_\mathrm{max} = \max_{i \in \mathcal{N}_\mathrm{P}} w_i.$$

The second-step minimization of the maximum play-out delay among peers can be formulated as problem P_2 by slightly modifying P_1, i.e., the objective function is replaced with

$$\min w_\mathrm{max}$$

and the following constraint is added.

$$\frac{1}{N_\mathrm{P}} \sum_{i \in \mathcal{N}_\mathrm{P}} w_i = \overline{w}^*,$$

where $\overline{w}^*$ denotes the average play-out delay minimized by solving problem P_1.

3.4. *Third step: Minimization of the average file retrieving time*

By solving problem P_2, we may still obtain more than one optimal solution that achieves $\overline{w}^*$ and w^*_max. In such situations, finding out a simpler optimal piece flow will be important to obtain the knowledge to design a sophisticated TFT-based P2P streaming protocol. In addition, it has been pointed out that video watching time tends to be shorter than video length in case of video streaming services [32]. This indicates that there is a potential risk that other peers may disappear from the system during the service. Furthermore, in case of on-demand streaming, some controlling methods, e.g., skipping and fast-forwarding, suddenly require a piece of future play-out position. Shortening the file retrieving time can be one of the countermeasures against these risks. Considering the above points, we set the third objective to be the minimization of the average file retrieving time among peers:

$$\overline{\tau} = \sum_{i \in \mathcal{N}_\mathrm{P}} \tau_i,$$

where τ_i ($i \in \mathcal{N}_\mathrm{P}$) is the file retrieving time of peer i, which is equal to the length of the period where peer i is a leecher [31]:

$$\tau_i = \sum_{t=0}^{T} y_{t,i}.$$

The third optimization can be formulated as problem P_2 by slightly modifying, i.e., the objective function is replaced with

$$\min \overline{\tau}$$

and the following constraint is added.

$$\max_{i \in \mathcal{N}_\mathrm{P}} w_i = w^*_\mathrm{max},$$

where w^*_max is the optimal maximum play-out delay among peers, which is obtained by solving P_2. We also define $\overline{\tau}^*$ is the optimal average file retrieving time, which is obtained by solving P_3. The impact of the third step optimization will be demonstrated by numerical results (Section 4.3).

3.5. Peers' behavior after file retrieving

When peers finish retrieving all the pieces, they can freely determine whether they leave the system. In actual P2P file retrieving systems, many seeds tend to stay in the system during a relatively short time [33]. In order to examine how peers' behavior after file retrieval affects the system performance, we apply two extreme scenarios, i.e., *seed sojourn scenario* and *seed departure scenario*. In the seed sojourn scenario, all peers are altruistic and serve as seeds after finishing the file retrieval. On the contrary, the seed departure scenario is a severe one where all peers are selfish and leave the system immediately after the file retrieval. Since there is no contribution from seeds in the seed departure scenario, the analysis under the seed departure scenario seems to be more important than that under the seed sojourn scenario.

We do not need any additional constraint in the seed sojourn scenario. On the contrary, the seed departure scenario requires to add the following constraint to P_1, P_2, and P_3.

$$x_{t,k,i,j} \leq y_{t-1,i}, \quad \forall t \in \mathcal{T}^+, \forall k \in \mathcal{M}, \forall i \in \mathcal{N}_\mathrm{P}, \forall j \in \mathcal{N},$$

which indicates that seeds do not transfer any pieces to others.

4. Analysis of optimal piece flow through numerical results

In this section, analysis of optimal piece flow will be given through numerical results, which are obtained by sequentially solving P_1, P_2, and P_3, with an existing solver CPLEX on a server with 10-core Intel Xeon CPU 3.0 GHz and 64 GB memory.

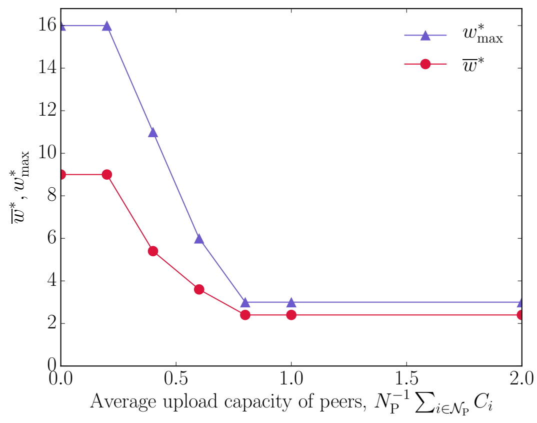

Figure 1: Impact of average upload capacity of peers ($N_\mathrm{P} = 5$, $M = 10$, $C_\mathrm{S} = 2$, with TFT, seed departure).

4.1. Evaluation model

As mentioned in Section 3.1, the proposed approach can be applied to both live streaming and on-demand streaming. In case of live streaming, peers tend to join the system at the same time. On the other hand, in case of on-demand streaming, each peer can enjoy the streaming service in an asynchronous fashion at steady state. However, if a popular file is released, a surge in peer arrival, called a *flash crowd*, will occur even in case of on-demand streaming [34, 35]. Under such a flash crowd, the upload capacity of server tends to be the bottleneck. Since it is important to reveal the potential of TFT-based P2P streaming, we consider the severest flash crowd, where all peers simultaneously join the system without any piece and start the streaming service. Note that the live streaming can support both stored media ($Q = M$) and live media ($1 \leq Q < M$) while the on-demand streaming can only support stored media ($Q = M$). In what follows, we mainly use the stored-media case ($Q = M$) and the impact of Q on system performance will be discussed in Section 4.7.

Since the objective is to analyze the optimal piece flow to discover the mechanism yielding it, a relatively small system is considered, i.e., one server S provides five peers, A, B, C, D, and E, with a streaming content composed of ten pieces ($N_\mathrm{D} = 1, N_\mathrm{P} = 5$, and $M = 10$). The validity of small-scale systems for analysis will be discussed in Section 4.2. The seed departure scenario is used as a default scenario because it is severer than the seed sojourn scenario. As for the upload capacities of server and peers, we have to find out universal nature of TFT-based P2P streaming but it is difficult to consider all possible patterns. In general, the upload capacity of server S becomes the bottleneck of streaming services because it will be not enough to provide a piece for every peer at every time, which requires $C_\mathrm{S} \geq N_\mathrm{P}$. To investigate the impact of the upload capacities of peers on the system performance and the roles of peers, we assume that the upload capacities of the peers are heterogeneous, i.e., $C_\mathrm{A} = C_\mathrm{B} = 3$, $C_\mathrm{C} = 2$, and $C_\mathrm{D} = C_\mathrm{E} = 1$, as in [31].

At first, we investigate how the average upload capacity of peers, $\overline{C}_\mathrm{P} = N_\mathrm{P}^{-1} \sum_{i \in \mathcal{N}_\mathrm{P}} C_i$, affects optimal average play-out delay $\overline{w}^*$ and optimal maximum play-out delay $w^*_{\max}$, when C_S is fixed to 2, as shown in Fig. 1. In Fig. 1, we decrease the upload capacity of a peer with the highest upload capacity one by one from the above default setting, so as to control the average upload capacity of peers. We can confirm that both $\overline{w}^*$ and $w^*_{\max}$ converge, that is the upload capacity of server S becomes the bottleneck, when the average upload capacity of peers is over 0.8. In what follows, as one of the server bottleneck cases, we set the default value of C_S to be 2.

Someone may also wonder whether the distribution of upload capacities of peers affects the system performance. To clarify this point, we compare $\overline{w}^*$ and $w^*_{\max}$ among three cases: above mentioned heterogeneous case, homogeneous case ($C_A = C_B = C_C = C_D = C_E = 2$), and biased case ($C_A = 6$, $C_B = C_C = C_D = C_E = 1$). Note that all the three cases have the same average upload capacity of peers. We obtained the same results, i.e., $\overline{w}^* = 2.4$ and $w^*_{\max} = 3.0$, among all these cases. In the succeeding evaluations, we use the scenario ($C_S = 2, C_A = C_B = 3$, $C_C = 2, C_D = C_E = 1$) as the default, and keep the average upload capacity among peers and distribution of upload capacities among peers even when we change N_P.

4.2. Validity and complexity of formulation

Since all the above mentioned problems, P_1, P_2, and P_3, are ILP, they are NP-hard. CPLEX solves ILP using a general and robust algorithm based on branch-and-cut algorithm [36]. When CPLEX solves ILP, it builds a tree with the linear relaxation of the original ILP, i.e., linear programming (LP) relaxation, at the root and sub-problems to optimize at the nodes of the tree. Note that the LP relaxation is LP that has the same objective function and same constraints as the original ILP but each integer variable is replaced by a continuous variable with the same lower and upper bounds. During the iterative solution search, CPLEX tries to find out an optimal solution by updating the upper bound and lower bound of objective function. Upper bound v_{ub} of objective function is updated by the minimum objective function value among feasible solutions of original ILP, which were found so far. On the other hand, lower bound v_{lb} of objective function is updated by the best solution of LP relaxation, which were found so far. As a result, CPLEX can define the optimality (quality) of the best feasible solution as gap g between upper bound and lower bound, i.e., $g = v_{\mathrm{ub}} - v_{\mathrm{lb}}$.

CPLEX can set a tolerance on gap g to control the trade-off between optimality and computation time. When gap g falls below predefined threshold $g_{\max}$, CPLEX stops the optimization process and guarantees that the difference between objective function value of found feasible solution and optimal objective function value is bounded by $g_{\max}$. In this paper, the default value of $g_{\max}$ is set to be zero, and thus the obtained results are strictly optimal.

The proposed approach based on ILP can trace the optimal behavior of TFT-based P2P streaming in a piece-by-piece manner, which will help us understand it deeply (See the detail in the succeeding sections). However, this approach also has a potential drawback in terms of complexity (scalability). In each problem, i.e., P_1, P_2, and P_3, the number of decision variables $x_{t,k,i,j}$ ($t \in \mathcal{T}, k \in \mathcal{M}, i, j \in \mathcal{N}$) is TMN^2 and that of constraints is bounded by $O(TMN^2)$. Since $x_{t,k,i,j}$'s are binary variables, the search space exponentially grows with increase of system size. As a result, CPU time quickly increases with N_P as shown in Fig. 2. In Fig. 2, we can also confirm the trade-off between optimality and CPU time by changing gap tolerance $g_{\max}$. Note that CPLEX offers multi-threaded parallel optimization where the execution time can be shortened according to the number of CPU cores.

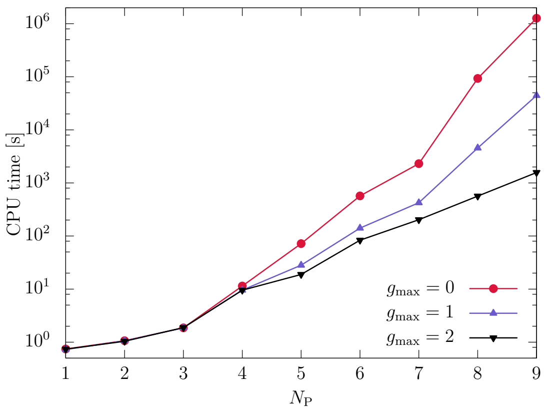

Figure 2: N_P vs. CPU time ($M = 10$, with TFT, seed departure).

Table 2: Impact of clustering structure on system performance.

	$N_P = 5$ ($C_S = 2$)	$N_P = 10$ ($C_S = 4$)	$N_P = 15$ ($C_S = 6$)
$\overline{w}^*$	2.4	2.3	2.3
$w^*_{\max}$	3.0	3.0	3.0

Fig. 2 shows that the proposed approach cannot directly reveal the behavior of large-scale systems. However, in what follows, we will show that the whole system can be divided into multiple isolated sub-systems, i.e., clusters, where peers can only communicate with others in the same cluster, without degrading the system performance, i.e., $\overline{w}^*$ and $w^*_{\max}$. Table 2 illustrates $\overline{w}^*$ and $w^*_{\max}$ for three systems: (1) $N_P = 5, C_S = 2$, (2) $N_P = 10, C_S = 4$, and (3) $N_P = 15, C_S = 6$. Note that system 2 (resp. system 3) is twice (resp. three times) as large as system 1 and system 1 can be regarded as one of clusters in system 2 (resp. system 3). We observe that $\overline{w}^*$ and $w^*_{\max}$ of system 1 is almost the same as those of systems 2 and 3, which indicates that, under the optimal piece flow, the whole system can be divided into multiple isolated clusters while keeping the system performance. In other words, the analytical results for a small-scale system can be applied to large-scale systems that consist of any number of the small-scale systems. The small-scale clustering structure will also be attractive to actual P2P protocols because each peer only requires to grasp a limited number of peers, which can contribute to reducing the communication and maintenance overhead.

4.3. Effect of three-step optimization

At first, the effect of three-step optimization is demonstrated. Figs. 3 through 5 show examples of the optimal piece flow ob-

t,k	1	2	3	4	5	6	7	8	9	10
1	▨ $\underline{A}_0^0$	🟦 $\underline{B}_0^2$								
2	B_0^2	A_0^2	🟦 $\underline{D}_0^3$	🟦 $\underline{C}_0^4$						
3	🟥 $C_0^4D_0^4\underline{E}_2^0$		A_2^0	B_2^0	🟦 $\underline{E}_0^4$					
4	■	🟥 $\underline{C}_2^0\underline{D}_2^0E_2^0$	B_2^0	A_2^0	A_2^0					
5		■	$C_2^0E_2^0$	D_2^0	B_2^0	🟦 $\underline{B}_0^2$	🟦 $\underline{C}_0^4$			
6			■	E_0^4	D_0^4	🟥 $\underline{A}_1^0$			🟦 $\underline{A}_0^3$	
7				■	C_0^3		A_0^3	🟦 $\underline{D}_0^3$	🟦 $\underline{E}_1^4$	
8					■	$C_1^4D_1^4E_1^4$	B_1^4	A_1^4	B_1^4	🟦 $\underline{E}_0^4$
9						■	$D_0^4E_0^4$	🟥 $\underline{B}_2^0C_2^0$		C_2^0
10							■	E_2^0	🟥 $\underline{D}_3^1$	▨ $A_3^1\underline{D}_2^2$
11								■	C_2^2	B_2^2
12									■	
13										■

Figure 3: Example of first-step optimal piece flow ($N_P = 5, M = 10$, with TFT, seed departure).

t,k	1	2	3	4	5	6	7	8	9	10
1	▨ $\underline{A}_0^0$	🟦 $\underline{B}_0^2$								
2	▨ $B_0^2\underline{D}_1^1$	A_1^1	🟦 $\underline{C}_0^3$							
3	🟥 $C_0^3\underline{E}_3^0$	C_3^0	$A_3^0D_3^0$	🟦 $\underline{E}_0^3$						
4	■	$D_0^3E_0^3$	B_0^3	A_0^3	🟦 $\underline{D}_0^3$	🟦 $\underline{B}_0^3$				
5		■	E_0^3	B_0^3	🟥 $\underline{A}_1^0$		🟦 $\underline{C}_0^4$			
6			■	$C_0^4D_0^4$	$B_0^4E_0^4$	A_0^4	A_0^4	🟦 $\underline{B}_0^3$	🟦 $\underline{B}_0^4$	
7				■	C_0^4	🟥 $\underline{E}_2^1$	$B_2^1D_2^1$	C_2^1		
8					■	$C_2^1D_2^1$	E_2^1	A_2^1	🟦 $\underline{D}_1^3$	🟦 $\underline{E}_0^4$
9						■			🟥 $\underline{A}_2^0\underline{E}_2^2$	
10							■	🟥 $\underline{D}_3^0E_3^0$		🟦 $A_3^0\underline{C}_1^2$
11								■	C_1^2	🟥 $B_1^2\underline{D}_3^1$
12									■	
13										■

Figure 4: Example of second-step optimal piece flow ($N_P = 5, M = 10$, with TFT, seed departure).

t,k	1	2	3	4	5	6	7	8	9	10
1	▨ $\underline{A}_0^0$	🟦 $\underline{B}_0^2$								
2	B_0^2	A_0^2	🟦 $\underline{E}_0^3$	🟦 $\underline{D}_0^4$						
3	🟥 $\underline{C}_2^0D_2^0E_2^0$		A_2^0	A_2^0	🟦 $\underline{C}_0^4$					
4	■	$C_0^4D_0^4E_0^4$	$B_0^4C_0^4$	B_0^4	$A_0^4B_0^4$	🟦 $\underline{E}_0^4$	🟦 $\underline{D}_0^5$			
5		■	D_0^5	E_0^5		A_0^5	C_0^5	🟦 $\underline{E}_0^5$	🟦 $\underline{D}_0^6$	
6			■	C_0^6	D_0^6		$B_0^6E_0^6$	C_0^6	A_0^6	🟦 $\underline{B}_0^5$
7				■	🟥 $\underline{E}_4^0$	🟥 $B_4^0\underline{C}_2^1$	A_2^1	$A_2^1B_2^1$	$B_2^1C_2^1E_2^1$	$A_2^1C_2^1D_2^1$
8					■	🟥 $\underline{D}_4^0$		D_4^0		E_4^0
9						■				
10							■			
11								■		
12									■	
13										■

Figure 5: Example of third-step optimal piece flow ($N_P = 5, M = 10$, with TFT, seed departure).

tained by solving P_1, P_2, and P_3, respectively. Note that there can be multiple solutions at each-step optimization and CPLEX gives us one of them, each of which has the optimal value of the corresponding objective function.

In these figures, (t, k)th cell includes information i_R^I about peer i retrieving piece k at time t, where I is the remaining time for peer i to play out piece k at the end of time t, i.e., $w_i + (k-1) - t$, and R is the number of peers that have piece k at the beginning of time t. I (resp. R) ranges from 0 to $w_i + (k-1)$ (resp. N_P). Smaller I (resp. R) indicates that the corresponding piece is more urgent (resp. rarer), and thus they show the trade-off between in-order policy and rarest-first policy in the optimal piece flow. In what follows, we assume three cases: in-order policy preferred case ($I < R$), rarest-first policy preferred case ($I > R$), and flat case ($I = R$). To focus on server's behavior, peer's id is underlined when the corresponding piece is directly received from server S and the corresponding cell is colored according to the preferred policy, i.e., light red (in-order policy preferred case), light blue (rarest-first policy preferred case), or light gray (flat case). Each black cell means the play-out timing of piece k for the peer with the maximum play-out delay. For example, in Fig. 3, $(3, 1)$th cell indicates that peer E directly receives piece 1 from S and peers C and D receive piece 1 from other peers at time 3, and peers C, D, and E has maximum play-out delay among peers, i.e., $w^*_{\max} = 3.0$. In addition, we can observe that server S gives preference to the in-order policy over the rarest-first policy because $(3, 1)$th cell is colored with light-red.

We observe that Fig. 3 is similar to Fig. 4 but Fig. 5 shows different characteristic compared with others. Actually, all of these results show the same performance in terms of average play-out delay $\overline{w}$ and maximum play-out delay $w_{\max}$, i.e., $\overline{w} = \overline{w}^* = 2.4$, $w_{\max} = w^*_{\max} = 3.0$. However, Figs. 3 through 5 have different average file retrieving time $\overline{\tau}$, i.e., 10.4, 10.6, 7.4, respectively. This difference comes from the degree of contribution of peers' uploading. In Figs. 3 and 4, the number of pieces that server S directly sends to peers is 18 and 20. On the other hand, that becomes 14 in Fig. 5. This indicates that the optimal piece flow obtained by the third-step optimization can sufficiently utilize the peers' upload capacities.

In what follows, we will deeply analyze the characteristic of the optimal flow at the third-step optimization.

4.4. Analysis of optimal piece selection

Next, we focus on the optimal balance between in-order policy and the rarest-first policy. Comparing Fig. 5 with Figs. 3 and 4, we observe that introducing the third-step optimization can simplify the optimal piece flow and emphasize the server's strategy in terms of the balance between in-order policy and the rarest-first policy. From the viewpoint of streaming services, in-order policy is important because peers require pieces in ascending order of piece ids to shorten their play-out delay. In other words, pieces are prioritized by the play-out order. On the other hand, TFT constraint requires peers to have different pieces with others. As a result, TFT-based P2P streaming should correctly select the balance between in-order policy and the rarest-first policy.

In Fig. 5, we focus on the optimal piece flow from server S, which is highlighted by the above mentioned three kinds of colors. We can also find that the corresponding receiving peers, which are underlined in those cells. We first focus on the characteristics of pieces sent from server S. We observe that server S strategically sends pieces to peers, taking account of the balance between in-order policy and the rarest-first policy. At first, server S basically selects pieces to send in ascending order of piece id from the rarest pieces, e.g., pieces 1 and 2 at $t = 1$, piece 3 and 4 at $t = 2$, and piece 5 at $t = 3$. This dissemination of in-order and rarest pieces will encourage piece exchange among peers. On the contrary, server S also requires to emphasize in-order policy to support urgent peers that immediately require pieces to play out, e.g., piece 1 at $t = 3$ and piece 5 at $t = 7$. This contributes the reduction of maximum play-out delay among peers.

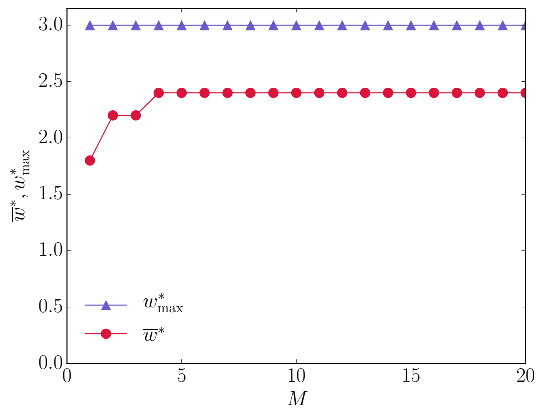

Figure 6: Impact of the number M of pieces ($N_P = 5$, with TFT, seed departure).

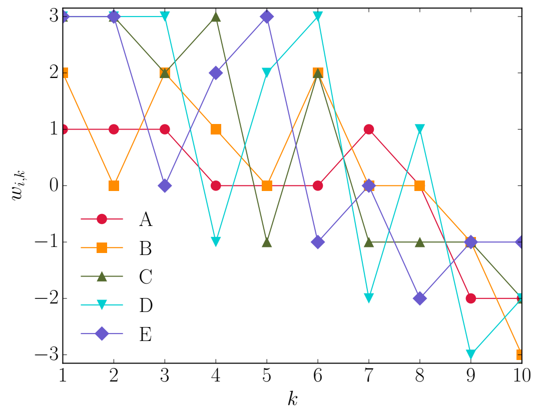

Figure 7: Peer i's waiting time $w_{i,k}$ for playing out piece k ($N_P = 5, M = 10$, with TFT, seed departure).

4.5. Analysis of optimal peer selection

Next, we focus on the characteristics of peers to which server S sends pieces. In Fig. 5, we observe that server S preferentially sends pieces to peers with higher upload capacities at the early stage where each peer has no piece, i.e., $t = [1, 3]$. In the succeeding stage, i.e., $t = [4, 10]$, server S preferentially supports piece retrieval of peers with lower upload capacities. Because of such piece dissemination form server S, the peers with higher upload capacities can exchange pieces with those with lower upload capacities.

4.6. Impact of system scale

In this section, we reveal how the system scale, i.e., the number M of pieces and the number N_P of peers, affects the performance of TFT-based P2P streaming.

We first focus on the impact of the number M of pieces. Fig. 6 illustrates the relationship between M and the system performance, i.e., $\overline{w}^*$ and $w^*_{\max}$, in case of $N_P = 5$. We first observe that optimal average play-out delay $\overline{w}^*$ gradually increases with M but converges to a certain value, i.e., 2.4 when $M = 4$. On the contrary, optimal maximum play-out delay $w^*_{\max}$ becomes constant, regardless of M.

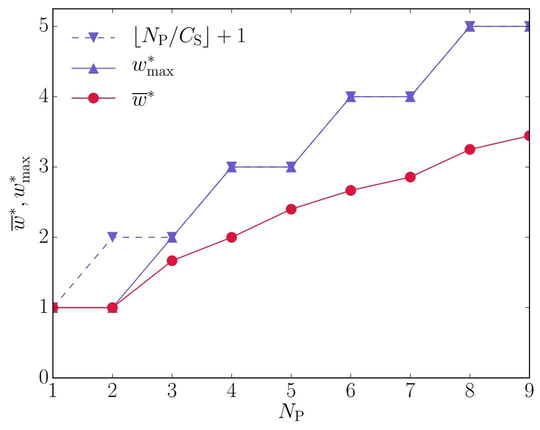

Figure 8: Impact of the number N_P of peers ($M = 10$, with TFT, seed departure).

To deeply analyze this characteristics, we also show peer i's waiting time $w_{i,k}$ for playing out piece k when $N_P = 5$, in Fig. 7. Recall that peer i's play-out delay is given by (3), i.e., $w_i = \max_{k \in \mathcal{M}} w_{i,k}$. In Fig. 7, we can find that every peer i has the maximum waiting time for playing out the first piece 1, i.e., $w_{i,1} = w_i$ ($\forall i \in \mathcal{N}_P$). This indicates that the fast retrieval of the first piece is important in TFT-based P2P streaming and dominates optimal maximum play-out delay $w^*_{\max}$. Note that Parvez et al. have also analyzed the maximum play-out delay in case of two variants of in-order policy and revealed similar results [17]. We also observe that $w_{i,k}$ tends to decrease with k when $k \geq 2$. This means that piece exchange among peers work well, which contributes to shorten optimal average play-out delay $\overline{w}^*$. As a result, we can conclude that the number M of pieces does not much affect the performance of TFT-based P2P streaming services.

Next, we focus on the impact of the number N_P of peers. Fig. 8 illustrates the relationship between N_P and the system performance, i.e., $\overline{w}^*$ and $w^*_{\max}$, in case of $M = 10$.

We first observe that $w^*_{\max}$ gradually increases with N_P while $w^*_{\max}$ increases in a step-by-step manner. $w^*_{\max}$ is also bounded by $\lfloor N_P/C_S \rfloor + 1$. Note that $\lceil N_P/C_S \rceil$ is equivalent to the time that server S directly sends a piece to all peers. Since the upload capacity of server S is bottleneck, this result indicates that peers' piece exchange works well in a cycle of $\lfloor N_P/C_S \rfloor + 1$, which can suppress the increase of $w^*_{\max}$.

We further confirm that this relationship is satisfied even when the upload capacity of server S changes. Fig. 9 shows the relationship between server S's upload capacity C_C and system performance, i.e., $\overline{w}^*$ and $w^*_{\max}$, in case of $N_P = 9$ and $M = 10$. We also show the upper bound of $w^*_{\max}$, $\lfloor N_P/C_S \rfloor + 1$. We can confirm that $w^*_{\max}$ follows $O(N_P/C_S)$ as in Fig. 8.

As a result, we can conclude that the number N_P of peers affects the performance of TFT-based P2P streaming services but $w^*_{\max}$ can be bounded by $O(N_P/C_S)$. Note that the upper bound of $w^*_{\max}$, $\lfloor N_P/C_S \rfloor + 1$, is also consistent with the results of Table 2 where all the three cases with the same $N_P/C_S = 2.5$ have $w^*_{\max} = \lfloor N_P/C_S \rfloor + 1 = 3.0$. As for the play-out delay, the lower bound of initial play-out delay has been mathematically

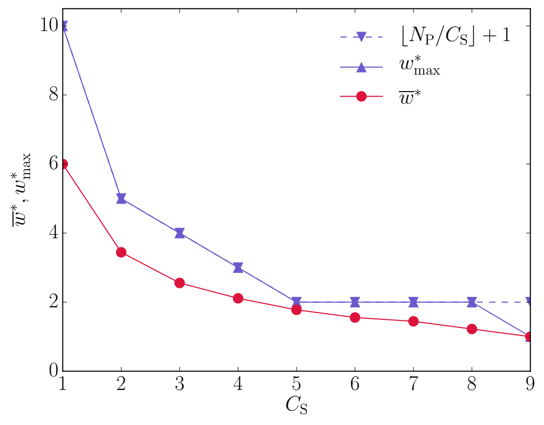

Figure 9: Impact of upload capacity C_S of server S ($N_P = 9, M = 10$, seed departure).

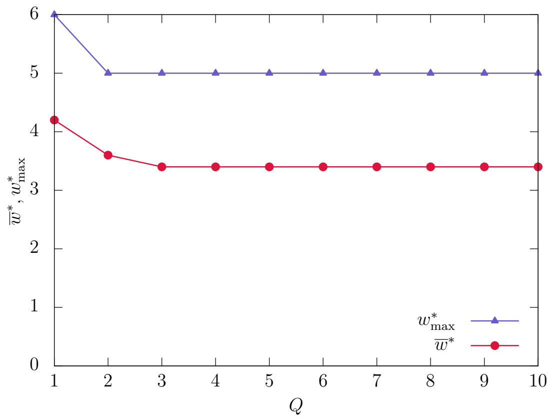

Figure 10: Impact of Q on $\overline{w}^*$ and $w^*_{\max}$ ($N_P = 9, M = 10$, seed departure).

analyzed to be $\lceil \log_{1+\overline{C}_P}(N_P/C_S) \rceil + 1$, when server S follows the in-order policy, all the N_P peers follow the newest-first policy, and there is no TFT constraint [29].

4.7. *Impact of media type*

In this section, we reveal how the media type, i.e., stored media and live media, affects the system performance, by changing range $[1, Q]$ of piece id that server S has at $t = 0$. Fig. 10 illustrates the impact of Q on $\overline{w}^*$ and $w^*_{\max}$ in case of $N_P = 9, M = 10$. As mentioned in Section 3.1, smaller Q will regulate the diversity of pieces among peers at the initial stage of streaming service. This makes the TFT-relationship among peers difficult, and thus both $\overline{w}^*$ and $w^*_{\max}$ in the strict live-media case ($Q = 1$) become worse than those in the stored-media case ($Q = M$). However, we observe that both $\overline{w}^*$ and $w^*_{\max}$ decrease with Q and converge at $Q = 3$, which is much smaller than the number of pieces, $M = 10$.

4.8. *Impact of relaxation of TFT constraint*

TFT constraint in (15) indicates that a pair of leechers should exchange the same number of pieces each other at time t. In this section, we reveal how the relaxation of TFT constraint affects the system performance. For this purpose, we extend

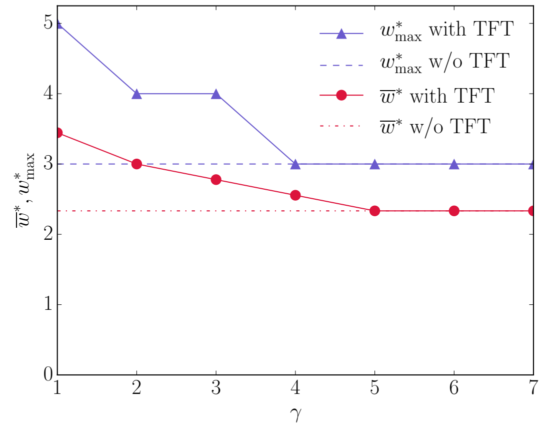

Figure 11: Impact of γ on $\overline{w}^*$ and $w^*_{\max}$ ($N_P = 9, M = 10$, seed departure).

t, k	1	2	3	4	5	6	7	8	9	10
1	$\underline{A}^0_0$	$\underline{B}^2_0$								
2	B^2_0	A^2_0	$\underline{E}^3_0$	$\underline{C}^4_0$						
3	$C^4_0\underline{D}^0_2E^0_2$	C^0_2	A^0_2	$A^0_2B^0_2$	$\underline{D}^4_0$					
4	■	$D^4_0E^4_0$	B^4_0		C^4_0	$\underline{E}^4_0$	$\underline{D}^5_0$			
5		■	$C^5_0D^5_0$	E^5_0	A^5_0	A^5_0	A^5_0	$\underline{B}^4_0$	$\underline{C}^6_0$	
6			■	$\underline{D}^0_4$	B^0_4	C^0_4	B^0_4	$A^0_4C^0_4D^0_4$	$A^0_4B^0_4$	$\underline{A}^4_0$
7				■	$\underline{E}^0_4$	$B^0_4\underline{D}^1_3$	C^1_3		$D^1_3E^1_3$	$B^1_3C^1_3D^1_3$
8					■		E^1_3	E^1_3		E^1_3
9						■				
10							■			
11								■		
12									■	
13										■

Figure 12: An example of third-step optimal piece flow ($N_P = 5, M = 10$, with TFT, seed sojourn).

(15) by introducing TFT window γ ($\gamma \geq 1$) during which TFT constraint among leechers should be satisfied:

$$\sum_{s=t-\gamma+1}^{t} \sum_{k \in \mathcal{M}} (x_{s,k,i,j} - x_{s,k,j,i}) \leq M \left(1 - y_{t-1,i} y_{t-1,j}\right),$$
$$t = \gamma, 2\gamma, \ldots, T, \forall i, j \in \mathcal{N}, \quad (17)$$

where T is assumed to be a multiple number of γ, which is equal or greater than $T_{\max}$ given by (4). Note that (17) with $\gamma = 1$ is equivalent to (15) and (17) with $\gamma = \infty$ means that there is no TFT constraint.

Fig. 11 shows the relationship between TFT window γ and system performance, i.e., $\overline{w}^*$ and $w^*_{\max}$, in case of $N_P = 9$ and $M = 10$. For comparison purpose, we also show the results in case of no TFT constraint, which can be regarded as the lower bound of those in case of TFT constraint. As we expected, both $\overline{w}^*$ and $w^*_{\max}$ decrease with γ and converge to the lower bound at 5 and 4, respectively. Although increase of γ can reduce $\overline{w}^*$ and $w^*_{\max}$, it also gives leechers chance to cheat the TFT constraint.

4.9. *Impact of peers' behavior after file retrieval*

Finally, we focus on the impact of peers' behavior after file retrieval. So far we evaluated the system performance in the seed departure scenario.

Fig. 12 illustrates an example of the third-step optimal piece flow in case of $N_P = 5, M = 10$, and seed sojourn scenario. Comparing Fig. 12 with Fig. 5, we observe that both results are similar, which indicates that peers' behavior after file retrieval does not much affect the system performance in TFT-based P2P streaming services. Actually, both results have the same $\overline{w}^*$ and $w^*_{\max}$.

5. Conclusion

In this paper, a discrete-time model of TFT-based P2P streaming was developed and ILP was formulated to determine the optimal piece flow where the average play-out delay was minimized. By solving the ILP using existing solver, i.e., CPLEX, we could obtain numerical examples of optimal piece flow. Through the analysis of obtained optimal piece flow, the following characteristics were revealed. (1) Optimal piece selection is based on the balance between in-order piece retrieving and the rarest-first piece retrieving. The server should basically distribute the rarest pieces in play-out order to encourage piece exchange among peers but also emphasize in-order policy to support urgent peers with lack of pieces to play out. (2) In terms of optimal peer selection, the server should send pieces to peers with higher upload capacities at the early stage, then support piece retrieval of peers with lower upload capacities. (3) The number of pieces does not affect the system performance. (4) The maximum play-out delay can be bounded by the ratio of the number of peers to the server's upload capacity. (5) How the relaxation of TFT constraint can improve the system performance.

Acknowledgments

This research was partly supported by JSPS KAKENHI Grant Number 15K00126 and KDDI Foundation Research Grant Program, Japan. The author thanks Prof. Tetsuya Takine and Mr. Youta Egusa for their valuable comments.

A. Linearization of products of binary variables

If all variables are binary, their product of nonlinear expression can be transformed into the combination of linear expressions as follows [37].

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

is equivalent to 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 (15) can be linearized.

[1] YouTube, LLC, YouTube, `http://www.youtube.com/`.
[2] Hulu, LLC, Hulu, `http://www.hulu.com/`.
[3] NetFlix, Inc, NetFlix, `http://www.netflix.com/`.
[4] Cisco, The Zettabyte Era – Trends and Analysis, `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html`.
[5] D. Xu, M. Hefeeda, S. Hambrusch, B. Bhargava, On Peer-to-Peer Media Streaming, in: Proc. of the 22nd International Conference on Distributed Computing Systems, 2002, pp. 363–371.
[6] BitTorrent, Inc., BitTorrent Live, `https://btlive.tv/`.
[7] PPLive, Inc., PPTV, `http://www.pptv.com/`.
[8] B. Cheng, L. Stein, H. Jin, X. Liao, Z. Zhang, GridCast: Improving Peer Sharing for P2P VoD, ACM Transactions on Multimedia Computing, Communications, and Applications 4 (4) (2008) 1–31.
[9] S. Xie, G. Keung, B. Li, A Measurement of a Large-Scale Peer-to-Peer Live Video Streaming System, in: Proc. of Packet Video'07, 2007, pp. 153–162.
[10] S. Jun, M. Ahamad, Incentives in BitTorrent Induce Free Riding, in: Proc. of ACM SIGCOMM'05, 2005, pp. 116–121.
[11] M. Barbera, A. Lombardo, G. Schembra, M. Tribastone, A Markov Model of a Freerider in a BitTorrent P2P Network, in: Proc. of IEEE GLOBECOM 2005, 2005, pp. 985–989.
[12] M. Feldman, C. Papadimitriou, J. Chuang, I. Stoica, Free-Riding and Whitewashing in Peer-to-Peer Systems, IEEE Journal on Selected Areas in Communications 24 (5) (2006) 1010–1019.
[13] D. Hughes, G. Coulson, J. Walkerdine, Free Riding on Gnutella Revisited: The Bell Tolls?, IEEE Distributed Systems Online 6 (6).
[14] J. Mol, J. Pouwelse, D. Epema, H. Sips, Free-Riding, Fairness, and Firewalls in P2P File-Sharing, in: Proc. of P2P'08, 2008, pp. 301–310.
[15] BitTorrent, Inc., BitTorrent, `http://www.bittorrent.com/`.
[16] N. Parvez, C. Williamson, A. Mahanti, N. Carlsson, Analysis of BitTorrent-Like Protocols for On-Demand Stored Media Streaming, in: Proc. of ACM SIGMETRICS'08, 2008, pp. 301–312.
[17] N. Parvez, C. Williamson, A. Mahanti, N. Carlsson, Insights on Media Streaming Progress Using BitTorrent-Like Protocols for On-Demand Streaming, IEEE/ACM Transactions on Networking 20 (3) (2012) 637–650.
[18] Z. Ma, K. Xu, J. Liu, H. Wang, Measurement, Modeling and Enhancement of BitTorrent-Based VoD System, Computer Networks 56 (3) (2012) 1103–1117.
[19] L. D'Acunto, N. Chiluka, T. Vinkó, H. Sips, BitTorrent-Like P2P Approaches for VoD: A Comparative Study, Computer Networks 57 (5) (2013) 1253–1276.
[20] ILOG, IBM ILOG CPLEX Optimizer, `http://www.ibm.com/software/commerce/optimization/cplex-optimizer/`.
[21] T.-Y. Wu, W.-T. Lee, N. Guizani, T.-M. Wang, Incentive Mechanism for P2P File Sharing Based on Social Network and Game Theory, Journal of Network and Computer Applications 41 (2014) 47–55.
[22] Xin Kang, Yongdong Wu, Incentive Mechanism Design for Heterogeneous Peer-to-Peer Networks: A Stackelberg Game Approach, IEEE Transactions on Mobile Computing 14 (5) (2015) 1018–1030.
[23] F. Azzedin, M. Yahaya, Modeling BitTorrent Choking Algorithm Using Game Theory, Future Generation Computer Systems 55 (2016) 255–265.
[24] C. Wu, Z. Li, X. Qiu, F. C. M. Lau, Auction-Based P2P VoD Streaming: Incentives and Optimal Scheduling, ACM Transactions on Multimedia Computing, Communications, and Applications 8s (1) (2012) 1–22.
[25] B. Fan, D. G. Andersen, M. Kaminsky, K. Papagiannaki, Balancing Throughput, Robustness, and In-Order Delivery in P2P VoD, in: Proc. of Co-NEXT'10, 2010, pp. 10:1–10:12.
[26] C. Y. Chang, C. F. Chou, K. C. Chen, Content-Priority-Aware Chunk Scheduling over Swarm-Based P2P Live Streaming System: From Theoretical Analysis to Practical Design, IEEE Journal on Emerging and Selected Topics in Circuits and Systems 4 (1) (2014) 57–69.
[27] C. Yang, Y. Zhou, L. Chen, T. Z. Fu, D. M. Chiu, Turbocharged Video Distribution via P2P, IEEE Transactions on Circuits and Systems for Video Technology 25 (2) (2015) 287–299.
[28] C. Zhao, J. Zhao, X. Lin, C. Wu, Capacity of P2P On-Demand Streaming with Simple, Robust, and Decentralized Control, IEEE/ACM Transactions on Networking 24 (5) (2016) 2607–2620.
[29] C. Feng, B. Li, B. Li, Understanding the Performance Gap between Pull-Based Mesh Streaming Protocols and Fundamental Limits, in: Proc. of INFOCOM'09, 2009, pp. 891–899.
[30] S. Tewari, L. Kleinrock, Analytical Model for BitTorrent-Based Live Video Streaming, in: Proc. of CCNC'07, 2007, pp. 976–980.
[31] M. Hasegawa, M. Sasabe, T. Takine, Analysis of Optimal Scheduling in Tit-for-Tat-based P2P File Distribution, IEICE Transactions on Communications E97-B (12) (2014) 2650–2657.

[32] Y. Chen, B. Zhang, Y. Liu, W. Zhu, Measurement and Modeling of Video Watching Time in a Large-Scale Internet Video-on-Demand System, IEEE Transactions on Multimedia 15 (8) (2013) 2087–2098.

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

[34] L. D'Acunto, T. Vinko, J. Pouwelse, Do BitTorrent-Like VoD Systems Scale under Flash-Crowds?, in: Proc. of P2P'10, 2010, pp. 1–4.

[35] C. Carbunaru, Y. M. Teo, B. Leong, T. Ho, Modeling Flash Crowd Performance in Peer-to-Peer File Distribution, IEEE Transactions on Parallel and Distributed Systems 25 (10) (2014) 2617–2626.

[36] M. Padberg, G. Rinaldi, A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems, SIAM Review 33 (1) (1991) 60–100.

[37] D. Chen, R. G. Batson, Y. Dang, Applied Integer Programming, Wiley, 2010.