PAPER

P2P Streaming Method Based on Playback Deadline Using Linear Programming

Masaki Hanada¹ and Hidehiro Kanemitsu²

¹Department of Information Systems, Tokyo University of Information Sciences
4-1 Onaridai, Wakaba-ku, Chiba-shi, Chiba 265-8501, Japan
2Global Education Center, Waseda University, 1-6-1 Nishiwaseda, Shinjuku-ku, Tokyo 169-8050, Japan
E-mail: mhanada@rsch.tuis.ac.jp, kanemih@ruri.waseda.jp

Abstract In this study, we propose a Peer-to-Peer (P2P) Video-on-Demand (VoD) streaming method which adjusts the receiving rate of pieces by considering the playback deadline of the pieces to reduce interrupted playback. P2P VoD streaming methods based on BitTorrent, such as BitTorrent Streaming (BiToS) and BiToS + Immediacy and Scarcity (BIS), have been proposed to support streaming. In the piece selection strategies of these two methods, a peer probabilistically selects a piece considering immediacy and/or scarcity (i.e., rarity) and receives the selected piece to reduce interrupted playback. However, the piece selection mechanisms still cause interrupted playback owing to probabilistic piece selection. In our proposed P2P VoD streaming method, a peer sequentially selects a piece, starting from the first piece and receives the selected piece at a transfer rate in time for predicted playback start time. As a result, the interrupted playback can be reduced further. In evaluation simulations, we show that the proposed streaming method outperforms previous streaming methods in terms of media playback continuity.

Keywords: P2P VoD streaming, bittorrent, deadline, linear programming

1. Introduction

Recently, Video-on-Demand (VoD) streaming services have been growing in number and size, as computing power and network bandwidth increases. Using VoD streaming services, users can play multimedia content, such as video and audio, while downloading it. When a client–server architecture is used for VoD streaming services, this leads to a degradation in quality of multimedia content because the content delivery server becomes overwhelmed with requests. Peer-to-Peer (P2P) technologies have received significant attention for VoD streaming services to solve the issues of client–server architecture. Various P2P mechanisms, such as Napster, Gnutella, and BitTorrent [1, 2, 3] have been developed and are widely used. While these mechanisms can support the distribution of time-insensitive content, they cannot easily support streaming because they are not designed for the distribution of time-sensitive content.

Conventionally, P2P VoD streaming systems can be generally classified into two categories: tree-based and mesh-based [4, 5, 6]. Tree-based overlays employ a tree distribution graph. The tree-based systems distribute video by pushing data from a peer to its children peers. P2Cast [7] and P2VoD [8] are examples of tree-based P2P VoD streaming systems. On the other hand, Mesh-based overlays are not confined to a static topology and are based on a mesh distribution graph. A peer dynamically connects to other peers to upload or download pieces. Many mesh-based P2P VoD streaming systems are based on BitTorrent because BitTorrent is a rapid and scalable file distribution scheme. BitTorrent Streaming (BiToS) [9], BiToS + Immediacy and Scarcity (BIS) [10], BitTorrent Assisted Streaming System (BASS) [11], BitTorrent Extensions for Streaming (BEST) [12, 13] and Distance-Availability Weighted Piece Selection Method (DAW) [14] are examples of BitTorrent-based P2P VoD streaming systems. In addition to these systems, PPLive P2P VoD system built and deployed by PPLive [15] and P2P VoD system considering group of pictures (GOPs) [16] are cited as examples using mechanisms which are not directly related to BitTorrent. Details of these mesh-based P2P VoD streaming systems are further described in Section 3.

In this paper, we focus on BitTorrent-based P2P VoD streaming systems because BitTorrent is a rapid and scalable file distribution P2P system and one of
the most popular mechanisms based on P2P technology. In addition, BitTorrent-based P2P VoD streaming systems have been proposed and evaluated in many previous studies.

BitTorrent is one of the most popular mechanisms of distributing files based on P2P technology. The main idea of BitTorrent is to relieve a server of heavy requests from an increase in the number of users by dividing the files into smaller pieces. A piece that is downloaded by a peer is distributed for uploading by the peer. Each peer needs to decide which piece should be downloaded, and from which peer those pieces should be downloaded. The performance of file distribution mainly depends on the piece and peer selection strategies. The piece selection strategy decides which piece to download, and the peer selection strategy decides from which peers to download (to give which peers permission to upload). In BitTorrent, the rarest first and choke algorithms have been introduced as the piece and peer selection strategies, respectively. In real-time playback of streaming, pieces are played while they are being downloaded. If the playback reaches a point where the piece has not yet been downloaded, it must wait for the piece to be downloaded. Therefore, the piece selection strategy is one of important elements for supporting real-time playback and needs to select and download pieces that will be played soon. BitTorrent-based P2P VoD streaming methods mainly focus on the piece selection strategy (i.e., the rarest first algorithms) of BitTorrent and extend this piece selection strategy to support real-time playback.

We focus on both BiToS and BIS and describe them because BiToS and BIS can provide high quality of real-time playback among BitTorrent-based P2P VoD streaming systems [9, 10]. BiToS is a method with the ability to support streaming. The piece selection strategy of BiToS prefers pieces that will be played soon. This piece selection strategy divides all the pieces of a streaming file into two sets. One set includes the earlier parts of the streaming file and the other set includes the later parts. A peer selects a set from the two sets with a probability and then selects a piece from the selected set according to the rarest first algorithm. BIS enhances BiToS to improve the quality of real-time playback of BiToS. Similar to BiToS, the piece selection strategy of BIS divides all the pieces of a streaming file into two sets: one set includes the earlier parts of the streaming file and the other set includes the later parts. A peer selects a set from the two sets with a probability and then selects a piece considering its immediacy and scarcity. Specifically, when a set which includes the earlier parts is selected, a piece is selected by weighting immediacy and scarcity from the selected set in BIS, while a piece is selected according to the rarest first algorithm in BiToS. As a result, BIS achieves high quality of real-time playback, compared with BiToS.

As mentioned above, the piece selection strategies of both BiToS and BIS use a probability to select a set from the two sets. By using a probability, there is a possibility to select a later piece that will not be played soon. The strategies that use probabilistic piece selection cause interrupted playback.

In this paper, we propose a P2P VoD streaming method based on the earliest first strategy with download speed adjustment to solve the issues. The piece selection strategy of the proposed P2P VoD streaming method selects a piece that is closest to the current playback piece from pieces that have not been downloaded yet (hereinafter referred to as the earliest first strategy). In addition, the proposed P2P VoD streaming method adjusts the download speed of the selected piece to avoid interrupted playback. Using these two functions, it is expected that the proposed method outperforms BiToS and BIS in terms of supporting smooth real-time playback.

The rest of the paper is organized as follows. In Section 2, we describe our assumed P2P streaming model. In Section 3, we describe related works, including BiToS and BIS. In Section 4, we present our proposed P2P VoD streaming method. In Section 5, we compare the proposed P2P VoD streaming method with BiToS and BIS. In Section 6, we consider the proposed P2P VoD streaming method from a viewpoint of practical use. Finally, Section 7 presents concluding remarks and the directions of our future work.

2. P2P Streaming Model

We assume a hybrid P2P streaming model, as shown in Fig.1. This model is based on BitTorrent, BiToS and BIS.

We suppose that a central server is located in the P2P network. The central server manages all the pieces in each peer. Here, pieces are numbered from the beginning to the end of the streaming file. The i-th piece is denoted by piece \( i \) \((i = 1, ..., N)\). Additionally, peers are numbered in the order in which they join the P2P network. The j-th peer is denoted by peer \( j \) \((j = 1, ..., M)\). The P2P network has a peer with all the pieces of the streaming file, denoted by peer 0.

The piece and peer selection process in the P2P streaming model is as follows: When a peer joins the P2P network or finishes downloading a piece, it connects to the server and obtains the set of pieces in each peer. The peer selects a piece from the set (i.e., the piece selection strategy) and downloads the selected piece from peers which hold the piece (i.e., the peer selection strategy) in parallel. After downloading the piece, the peer plays the downloaded piece. The downloaded piece can then be downloaded by the other peers.

In Fig.1, peer 0 has all the pieces of the streaming
file. Peer 1 has pieces 1 and 2, and is downloading piece 3. Peers 2 and 3 can download pieces 1 and 2 from peer 1 if peer 1 gives peers 2 and 3 permission to upload. Peers 2 and 3 hold piece 1, and are downloading piece 2. Peer 1 can download pieces 1 and 2 from peers 2 and 3 if peers 2 and 3 give peer 1 permission to upload. In this case, peer 1 has held pieces 1 and 2 and does not download them.

3. Related Works

3.1 BiToS

BiToS is a P2P streaming method based on BitTorrent. The piece selection strategy of BiToS prefers pieces that will be played soon. In BiToS, all the pieces of a streaming file that have not yet been downloaded are divided into two sets (High Priority Set and Remaining Pieces Set). The two sets and the piece selection strategy of BiToS are shown in Fig.2. The two sets are:

- High Priority Set: Contains the pieces of a streaming file that have not yet been downloaded and will soon be played by the player. This set has a fixed size of pieces and this size is a system parameter.
- Remaining Pieces Set: Contains the pieces of a streaming file that have not yet been downloaded and are not in High Priority Set.

The piece selection process of BiToS is as follows:

**Step 1:** When a peer requests a streaming file, the peer connects to the central server and obtains information such as the list of pieces that each peer has and the connection status of each peer.

**Step 2:** The peer divides all the pieces that have not yet been downloaded into two sets (High Priority Set and Remaining Pieces Set).

**Step 3:** The peer selects High Priority Set with the probability $p$ and Remaining Pieces Set with the probability $1 - p$. The probability $p$ is a system parameter.

**Step 4:** If High Priority Set is selected in Step 3, the peer selects a piece from High Priority Set according to the rarest first algorithm. The peer moves the piece that has the lowest piece number in Remaining Pieces Set to High Priority Set to keep a fixed size of High Priority Set.

**Step 5:** If Remaining Pieces Set is selected in Step 3, the peer selects a piece from Remaining Pieces Set according to the rarest first algorithm.

**Step 6:** The peer starts to download the chosen piece from peers that hold the piece.

In the example of Fig.2, High Priority Set is selected by using the probability $p$, and then piece 10 is selected according to the rarest first algorithm.

In BiToS, the larger the probability $p$, the higher the possibility of selecting a piece that will be played soon.

3.2 BIS

BIS enhances BiToS to improve the quality of real-time playback of BiToS. In BIS, a peer selects a piece considering the *immediacy* and *scarcity* of each piece. All the pieces of a streaming file that have not yet been downloaded are divided into two sets (High Priority Set and Remaining Pieces Set), in the same way as BiToS.

Steps 1, 2, 3, 5, and 6 of the piece selection process of BIS are the same as those of BiToS. However, Step 4 of the BIS piece selection process is different from that of BiToS. The piece selection strategy of BIS is shown in Fig.3.

**Step 4** of the piece selection process of BIS is as follows:

**Step 4:** If High Priority Set is selected in Step 3, the peer calculates the *importance degree* ($D_i$) of each piece and selects the piece that has the highest importance degree ($D$).

The importance degree of piece $i$, $D_i$, is defined as:

$$D_i = cI_i + (1 - c)S_i$$  \hspace{1cm} (1)

where $I_i$ and $S_i$ are the immediacy and scarcity degrees.
of piece $i$, and $c$ is the weight coefficient ($c$ is a system parameter).

The immediacy degree of piece $i$, $I_i$, is defined as:

$$I_i = \begin{cases} 0 & (i = 0, ..., P) \\ 1 - (i - h)/P & (i = h, ..., P) \\ 1 & i \in DP \end{cases}$$

where $h$ is the next piece to be played, $P$ is the previously played piece, and $DP$ is the set of pieces that have not been downloaded.

The scarcity degree of piece $i$, $S_i$, is defined as:

$$S_i = \frac{(N - m)}{N}$$

where $N$ is the total number of peers in the P2P network and $m$ is the number of peers that have piece $i$.

In the example of Fig.3, High Priority Set is selected by using the probability $p$, and then piece 7 is selected by using the importance degree $D$.

Similar to BiToS, the higher the probability $p$, the higher the probability of selecting a piece that will be played soon (i.e., selecting a piece in High Priority Set). In addition, when High Priority Set is selected, a piece is selected according to the rarest first algorithm in BiToS. Consequently, the possibility of selecting a piece that will be played soon is higher than that in BiToS.

3.3 Mechanism of playback waiting time and playback interruption time

An example of the playback waiting time and the playback interruption time is shown in Fig.4. The playback waiting time is a period between the request time for the first piece (piece 1) and the playback starting time of it. The playback interruption time is a period between the playback finishing time of a piece and the starting time of the next piece. This time occurs when a piece has not yet been downloaded by the playback finishing time of the previous piece. That is, piece $i$ should have been downloaded by the playback finishing time of piece $i - 1$ to avoid interrupted playback. Hereinafter, the playback finishing time of piece $i - 1$ is referred to as the Playback Deadline of piece $i$. The upper part of Fig.4 shows the behavior of playback of pieces and the lower part shows the behavior of the download of pieces. In Fig.4, the playback interruption of piece 3 occurs because piece 3 has not been downloaded by the playback finishing time of piece 2.

3.4 Problem of BiToS and BIS

The piece selection strategies of BiToS and BIS select a piece by using the probability $p$. These probabilistic piece selection strategies cause an increase in playback waiting time and playback interruption time, even though the available bandwidth exists. As shown in Fig.2 and Fig.3, we consider the case when piece 1–6 have been played, and piece 8–9, piece 11 and piece 19 have been downloaded, and pieces 10 and 7 are respectively selected under BiToS and BIS. Assuming that piece 6 is selected and downloaded after that, the playback waiting time and the playback interruption time under BiToS and BIS are shown in Fig.5. In the middle part of Fig.5, the playback interruption occurs because piece 6 has not been downloaded by the playback finishing time of piece 5. Similarly, in the lower part of Fig.5, the playback interruption occurs because piece 6 has not been downloaded by the playback finishing time of piece 5. As mentioned above, there is a possibility that the probabilistic piece selection strategies cause an increase in playback waiting time and playback interruption time.

3.5 Other P2P streaming methods

In this section, we first describe BitTorrent-based P2P VoD streaming methods except for BiToS and BIS. BitTorrent Assisted Streaming System (BASS) [11] is a method that uses the rarest first algorithm to select a piece, similar to BitTorrent. In BASS, if a piece has not been downloaded by its playback deadline, the
piece is downloaded from the content delivery server according to client–server architecture. As a result, the load on the content server is high and the playback interruption time may be large. In BitTorrent Extensions for Streaming (BEST) [12, 13], all the pieces of a streaming file that have not yet been downloaded are divided into two sets (Playback Buffer and Picking Range). The playback buffer includes the earlier parts of the streaming file and the picking range includes the later parts. BEST selects a piece from the playback buffer according to the earliest first strategy if the playback buffer is not empty. If the playback buffer is empty, BEST selects a piece from the picking range according to several piece selection algorithms.

In the performance evaluation, the earliest first strategy for selecting a piece from the picking range showed good performance in terms of the playback deadline miss ratio. Distance-Availability Weighted Piece Selection Method (DAW) [14] compromises between the earliest first and rarest first strategies. In DAW, all the pieces of a streaming file that have not yet been downloaded are divided into two sets, similar to BEST. DAW selects a piece from the playback buffer according to the earliest first strategy if the playback buffer is not empty. If the playback buffer is empty, DAW calculates the priority $\text{Priority}_r = 1/((P_r - P_c) \times m_r)$, where $P_r$ is the sequence number of piece $r$, $m_r$ is the number of peers who hold piece $r$, and $P_c$ is the sequence number of the current last piece in the playback buffer. In the performance evaluation, this method provided a better solution than either the earliest first strategy or the rarest first strategy. In [17], the authors proposed a mixed strategy of the rarest first and greedy to propagate the chunks (i.e., pieces) as many peers as possible so as to maximize playback continuity from a theoretical viewpoint.

Next we describe P2P VoD streaming methods except for BitTorrent-based P2P VoD streaming methods. P2Cast [7] and P2VoD [8] are the tree-based P2P VoD streaming systems. In P2Cast, the entire video stream is delivered over the application-level multicast tree. Peers not only receive the requested video stream, but also contribute to the overall VoD service by forwarding the video stream to other peers. In P2VoD, requests from peers are handled by utilizing the peer’s resources. Each peer has a FIFO buffer to cache the most recent content of the video stream it receives. Existing peers forward the stream to a new peer as long as they have enough bandwidth and still hold the first block of the video file in the buffer. In addition to P2P VoD streaming methods, many P2P live streaming methods were proposed and developed in previous studies. Chainsaw [18], DONet [19] and PRIME [20] are examples of P2P live streaming systems. In this paper, we focus on P2P VoD streaming methods and please refer to [18], [19] and [20] for more extensive descriptions of P2P live streaming systems.

In [10], BIS showed that it outperformed BiToS, BEST, and DAW in terms of the playback interruption time if the system parameters (i.e., the probability ($p$), weight between the immediacy and scarcity ($c$), and size of High Priority set ($k$)) are appropriately set. Therefore, we compare our proposed method with only BIS in terms of the playback interruption and waiting times in Section 5.

4. Proposed P2P VoD Streaming Method

4.1 Overview

As mentioned in Sections 1 and 3, the probabilistic piece selection strategies of BiToS and BIS increase the playback waiting time and interruption time. The earliest first strategy is the best solution for reducing the playback interruption time. However, the earliest first strategy reduces the overall availability of pieces because a lot of rare pieces exist in the network. In order to mitigate the decrease in the overall availability of pieces, the proposed P2P VoD streaming method utilizes the available bandwidth and efficiently distributes pieces to the whole network from the earlier parts of the streaming file. The proposed method has the following two functions:

- Earliest First Strategy: A peer selects a piece that is the closest to the current playback piece from pieces that have not yet been downloaded to reduce the playback interruption time. Correctly, a peer selects pieces in the ascending order of the piece number.
- Download Speed Adjustment Function: In order to utilize available bandwidth and to efficiently distribute pieces to the whole network, the download speed of a piece is adjusted to meet to the playback deadline of the piece.
4.2 Piece selection strategy and download speed adjustment

The proposed method uses the earliest first strategy as the piece selection strategy and has the download speed adjustment function. For the download speed adjustment, the central server manages a list of peers in the P2P network and a set of pieces in each peer. Then, it calculates the download speed of the downloading piece or the next piece to be downloaded for each peer and notifies each peer of this.

A scenario example of the proposed method is shown in Fig. 7. The piece selection strategy and the download speed adjustment function of the proposed method are as follows (the following step numbers correspond with the step number of Fig. 7):

**Step 1:** When a peer requests a streaming file after joining a P2P network, the peer selects piece 1 (i.e., the first piece). Additionally, when a peer has downloaded the selected piece, the peer selects a piece according to the earliest first strategy. This step is the piece selection strategy of the proposed method. When a peer decides the next piece to be downloaded, the peer connects to the central server and notifies the central server of this.

**Step 2:** The central server requests the download speed at which each peer downloads the next download piece or the current downloading piece to meet its playback deadline from each peer. When a peer receives this request, it calculates the download speed $R_{pj}^p$ of piece $p_j$.

$$R_{pj}^p = \frac{DL_{pj}^p}{SP_{pj}^p} \tag{3}$$

where $DL_{pj}^p$ is the remaining size of the downloading piece $p_j$ and $SP_{pj}^p$ is a period between the current time and the estimated playback time of the downloading piece $p_j$. Hereinafter, $SP_{pj}^p$ is referred to as Slack Time. An example of the slack time is shown in Fig. 8.

**Step 3:** The central server receives the required download speeds from all the peers and calculates an assigned download speed for each peer using linear programming in order to utilize available bandwidth. The details of the assigned download speeds will be described in Section 4.3.

**Step 4:** The central server notifies all the source peers of the assigned download speeds.

**Step 5:** Each peer that receives the assigned download speed sends (or uploads) the next piece to the destination peers at the assigned download speed. **Step 2** to **Step 5** are the download speed adjustment function of the proposed method.

4.3 Calculation of download speed

In this section, we describe the following two calculations. The first is the calculation of the download speed at which each peer downloads the next piece or the current downloading piece to meet its playback deadline, as described in **Step 2** of Section 4.2. The second is the calculation of the assigned download speed for each peer using linear programming to utilize available bandwidth, as described in **Step 3** of Section 4.2.

We first describe the former calculation. Let us denote a current downloading piece of peer $j$ by piece $p_j$ and denote a current playback piece by piece $c_j$. As mentioned in **Step 2** of Section 4.2, the central server requests the download speed at which each peer downloads the next downloading piece or the current downloading piece to meet its playback deadline from each peer. When peer $j$ receives this request, it calculates the download speed $R_{pj}^p$ of piece $p_j$.

$$R_{pj}^p = \frac{DL_{pj}^p}{SP_{pj}^p} \tag{3}$$

where $DL_{pj}^p$ is the remaining size of the downloading piece $p_j$ and $SP_{pj}^p$ is a period between the current time and the estimated playback time of the downloading piece $p_j$. Hereinafter, $SP_{pj}^p$ is referred to as Slack Time. An example of the slack time is shown in Fig. 8.
The slack time $SP_{ij}^p$ is defined as:

$$SP_{ij}^p = R_{ij}^p + \left(\frac{p_j - c_j - 1}{BR}\right) \times L$$

(4)

where $L$ and $BR$ are the piece size and the playback bitrate of each piece, respectively. If peer $j$ downloads piece $p_j$ at the download speed $R_{ij}^p$, the playback interruption does not occur. At the end of Step 2 of Section 4.2, each peer $j$ replies the download speed $R_{ij}^p$ to the central server.

We next describe the latter calculation. As mentioned in Step 3 of Section 4.2, the central server receives the download speed, $R_{ij}^p$, from all the peers and calculates an assigned download speed for each peer using linear programming to utilize the available bandwidth. The problem of the assigned download speed of the downloading piece $p_j$ of peer $j$ can be formulated as a linear programming problem, as follows:

maximize $\sum_{i \in AP} \sum_{j \in AP} A_{ij}^p$ (5a)

subject to $\sum_{i \in PI_j} A_{ij}^p + \sum_{p_i \in PE_j} A_{ij}^p \leq B_j$, $\forall j \in AP$ (5b)

$\sum_{i \in AP} A_{ij}^p \geq R_{ij}^p$, $\forall j \in AP$ (5c)

Here, $A_{ij}^p$ is the assigned download speed from peer $i$ to peer $j$ to meet the playback deadline of the downloading piece $p_j$ for peer $j$. $B_j$ is the maximum bandwidth of peer $j$, $AP$ is the set of peers that exists in the P2P network, $PI_j$ is the set of peers that hold the downloading piece $p_j$ of peer $j$, and $PE_j$ is the set of pieces that peer $j$ holds.

We explain each subject below. The subject function (5b) means that the total upload and download speed between peer $j$ and other peers must be less than the maximum bandwidth of peer $j$. $B_j$. The subject function (5c) means that the playback interruption of the downloading piece $p_j$ of peer $j$ does not occur if the piece $p_j$ is downloaded at a speed more than $R_{ij}^p$.

In objective function (5a), the total download speed,
4.2, peers 1, 2, and 3 reply that piece 3 of peer 1 should be downloaded from peer 0 at a speed faster than or equal to $\frac{4}{3}$ (Mbps) to meet the playback deadline. Similarly, (6b) means that piece 2 of peer 3 should be downloaded from peers 0 and 1 at a speed faster than or equal to $\frac{8}{5}$ (Mbps), and (6c) means that piece 2 of peer 3 should be downloaded from peers 0 and 1 at a speed faster than or equal to 4 (Mbps).

As shown in Fig. 1, the total upload and download speed between peer $j$ and other peers must be less than the maximum bandwidth of peer $j$, $B_j$. With the maximum bandwidth $B_j (j = 0, 1, 2, 3)$ of 8 (Mbps), the subject function (5b) is as follows:

$$A_{0,1}^3 + A_{0,2}^3 + A_{0,3}^3 \leq 8 \text{ (Mbps)} \quad (6a)$$

$$A_{0,2}^3 + A_{1,2}^3 \geq \frac{8}{5} \text{ (Mbps)} \quad (6b)$$

$$A_{0,3}^3 + A_{1,3}^3 \geq 4 \text{ (Mbps)} \quad (6c)$$

(6a) means that piece 3 of peer 1 should be downloaded from peer 0 at a speed faster than or equal to $\frac{4}{3}$ (Mbps) to meet the playback deadline. Similarly, (6b) means that piece 2 of peer 2 should be downloaded from peers 0 and 1 at a speed faster than or equal to $\frac{8}{5}$ (Mbps), and (6c) means that piece 2 of peer 3 should be downloaded from peers 0 and 1 at a speed faster than or equal to 4 (Mbps).

As shown in Fig. 1, the total upload and download speeds between peer $j$ and other peers must be less than the maximum bandwidth of peer $j$, $B_j$. With the maximum bandwidth $B_j (j = 0, 1, 2, 3)$ of 8 (Mbps), the subject function (5b) is as follows:

$$A_{0,1}^3 + A_{0,2}^3 + A_{0,3}^3 \leq 8 \text{ (Mbps)} \quad (7a)$$

$$A_{0,1}^3 + A_{1,2}^3 + A_{1,3}^3 \leq 8 \text{ (Mbps)} \quad (7b)$$

$$A_{0,2}^3 + A_{1,2}^3 \leq 8 \text{ (Mbps)} \quad (7c)$$

$$A_{1,3}^3 + A_{1,3}^3 \leq 8 \text{ (Mbps)} \quad (7d)$$

(7a) means that focusing on peer 0, the total upload speed from peer 0 to peers 1, 2, and 3 must be less than or equal to 8 (Mbps) (i.e., the maximum bandwidth of peer 0). Similarly, (7b) means that focusing on peer 1, the sum of the download speeds from peer 0 to peer 1 and the upload speeds from peer 1 to peers 2 and 3 must be less than or equal to 8 (Mbps), (7c) means that focusing on peer 2, the total download speeds from peers 0 and 1 to peer 2 must be less than or equal to 8 (Mbps), and (7d) means that focusing on peer 3, the total download speeds from peers 0 and 1 to 3 must be less than or equal to 8 (Mbps).

Consequently, the following linear programming problem is defined:

maximize $A_{0,1}^3 + A_{0,2}^2 + A_{0,3}^2 + A_{1,2}^2 + A_{1,3}^2$ \quad (8a)

subject to $A_{0,1}^3 + A_{0,2}^3 + A_{0,3}^3 \leq 8 \text{ (Mbps)}$ \quad (8b)

$A_{0,1}^3 + A_{1,2}^3 + A_{1,3}^3 \leq 8 \text{ (Mbps)}$ \quad (8c)

$A_{0,2}^3 + A_{1,2}^3 \leq 8 \text{ (Mbps)}$ \quad (8d)

$A_{1,3}^3 + A_{1,3}^3 \leq 8 \text{ (Mbps)}$ \quad (8e)

$A_{0,1}^3 \geq \frac{4}{3} \text{ (Mbps)}$ \quad (8f)

$A_{0,2}^3 + A_{1,2}^3 \geq \frac{8}{5} \text{ (Mbps)}$ \quad (8g)

$A_{0,3}^3 + A_{1,3}^3 \geq 4 \text{ (Mbps)}$ \quad (8h)

We solve this linear programming problem by using lp_solve 5.5.2.0[21], which is a free linear programming solver and obtain the answers $A_{0,1}^3 = 1.000$ (Mbps), $A_{0,2}^3 = 0.000$ (Mbps), $A_{0,3}^3 = 7.000$ (Mbps), $A_{1,2}^3 = 6.000$ (Mbps), and $A_{1,3}^3 = 1.000$ (Mbps).

As shown in Fig. 7, in Step 4 of Section 4.2, the central server notifies $A_{0,1}^3 = 1.000$ (Mbps), $A_{0,2}^3 = 0.000$ (Mbps), $A_{0,3}^3 = 7.000$ (Mbps) of peer 0, and $A_{1,2}^3 = 6.000$ (Mbps) and $A_{1,3}^3 = 1.000$ (Mbps) of peer 1. Finally, in Step 5 of Section 4.2, piece 3 of peer 1 is downloaded from peer 0 at rate 1.000 (= $A_{0,1}^3$) (Mbps), piece 2 of peer 2 is downloaded from peer 0 at rate 0.000 (= $A_{0,3}^3$) (Mbps) and from peer 1 at rate 6.000 (= $A_{1,2}^3$) (Mbps), and piece 2 of peer 3 is downloaded from peer 0 at rate 7.000 (= $A_{0,3}^3$) (Mbps) and from peer 1 at rate 1.000 (= $A_{1,3}^3$) (Mbps).

5. Evaluation by Simulation Experiment

We compare the playback interruption and waiting times under the proposed method with only BIS through a simulation experiment to evaluate the proposed method. This is because BIS outperforms BiToS, BEST and DAW, as described in Section 3.4 [10].

In the proposed method, a central server calculates the download speed (i.e., $A_{0,j}^3$) by solving the linear programming problem in time for the playback deadline of each piece in Step 3 of Section 4.2. In general, the larger the number of variables and constraints of the linear programming problem is, the larger the processing time required for solving the linear programming problem is. If we have an assumption that the processing time is 0 in the simulation, there is a possibility that we cannot precisely evaluate the proposed method. Therefore, in advance of the simulation experiment, we measure the processing time required for solving the linear programming problem on a specific real computer that assumes a central server in a preliminary experiment and then use the exponential functions that are approximated by the measured processing time, as the processing time in the simulation experiment.
Table 1 Parameters in preliminary experiment

| Piece Count | 150 |
| Bitrate     | 2 (Mbps) |
| Piece Size  | 1 (Mbyte) |
| Peer Count  | 50 |
| Maximum Bandwidth | 8 (Mbps) |
| Average Arrival(Joining) Interval (µs) of Peers | 2, 3, 6, 8 (µs) |
| Simultaneous Connection Count | 4 |
| Buffering Time | 0, 4, 8, 12 (µs) |

Table 2 Approximated exponential functions

| Group | Constraint Count | Approximated Exponential Function (µs) | Coefficient of Determination (R²) |
|-------|------------------|---------------------------------------|----------------------------------|
| 1     | 1–50             | y = 143.06e^{0.06x}                   | 0.9543                           |
| 2     | 51–100           | y = 346.99e^{-0.06x}                  | 0.9607                           |

We describe the approximated exponential functions in Section 5.1, and show simulation results in Section 5.2.

5.1 Approximated exponential functions for processing time

We measure the processing time required for solving the linear programming problem on a specific real computer. The specifications of the real computer used in the preliminary experiment are: Core i7-3770 Processor and 16 GB Memory. The parameters of the preliminary experiment are shown in Table 1. These parameters are the same as those of the simulation experiment, as shown in Section 5.2. In the preliminary experiment, a peer joins the P2P network according to a Poisson process and then requests a streaming file. Each peer leaves the P2P network after finishing the playback of all the pieces of the streaming file. For more details, refer to the following.

The processing time required for solving the linear programming problem depends on the number of variables and constraints. The measured processing times obtained by the preliminary experiment are divided into 2 groups according to the number of constraints. In each group, by applying an exponential approximation to the measured processing time, we define the exponential function \( y = A e^{B x} \), where the x-axis is the number of variables and y-axis is the processing time (µs). The defined exponential functions in each group are shown in Table 2. The coefficients of determination (R²) of groups 1 and 2 are 0.9543 and 0.9607, respectively. These results imply a high level of goodness-of-fit model. We use these defined exponential functions as the processing time required for solving the linear programming problem on a central server in the simulation experiment. Additionally, the maximum processing time is 11404 (µs) in the preliminary experiment.

5.2 Simulation experiment and results

In this section, we compare the playback interruption and waiting times under the proposed method with only BIS through the simulation experiment.

The parameters of the simulation are shown in Table 3. In the simulation, peers join a P2P network according to a Poisson process with an average interval \( \lambda \). Each peer joins the P2P network and then soon requests a streaming file. Each peer leaves the P2P network after finishing the playback of all the pieces of the streaming file. For more details, refer to the following.

In the simulation scenario, we use a Poisson process to model the arrival of new peers. The parameters of the Poisson process are shown in Table 3. In the simulation, peers join the P2P network according to a Poisson process with an average interval \( \lambda \). Each peer joins the P2P network and then soon requests a streaming file. Each peer leaves the P2P network after finishing the playback of all the pieces of the streaming file. For more details, refer to the following.

The parameters of the simulation are shown in Table 3. In the simulation, peers join a P2P network according to a Poisson process with an average interval \( \lambda \). Each peer joins the P2P network and then soon requests a streaming file. Each peer leaves the P2P network after finishing the playback of all the pieces of the streaming file. For more details, refer to the following.

The parameters of the simulation are shown in Table 3. In the simulation, peers join a P2P network according to a Poisson process with an average interval \( \lambda \). Each peer joins the P2P network and then soon requests a streaming file. Each peer leaves the P2P network after finishing the playback of all the pieces of the streaming file. For more details, refer to the following.

We assume that the proposed method is used in small-scale P2P networks. In the simulation, the number of peers is 50 (including peer 0, which has all the pieces of the streaming file) and the number of pieces is 150. This is because, the larger the number of constraints of the linear programming problem is, the higher the work load of the central server is. The simultaneous connection count is 4 according to BitTorrent. In the intrinsic setting of the proposed method, we set \( R_f^j (\forall j \in AP) \) as the playback bitrate times two (i.e., 4 (Mbps)) where \( f \) means the first piece (i.e., piece 1). This is because it is desired that the first piece (i.e., piece 1) is quickly downloaded to some extent. The intrinsic parameters of BIS are the size of the high priority set (hereinafter referred to as parameter \( q \)), parameter \( p \), and parameter \( c \). Parameter \( q \) takes the three values 3, 9, and 12, while parameters \( p \) and \( c \) take the four values 1, 0.8, 0.6, and 0.4.

The approximated exponential functions of the pro-
Table 4 Playback waiting and interruption times (BIS)

| Arrival Interval (a (s)) | Playback waiting time (s) | Cumulation | Average (per peer) | Cumulation | Average (per peer) | Average (per piece) |
|--------------------------|---------------------------|------------|--------------------|------------|--------------------|--------------------|
| 2                        | 126.091                   | 2.574      | 749.673            | 15.299     | 0.102              | 11404              |
| 4                        | 110.738                   | 2.269      | 404.721            | 8.239      | 0.055              |                    |
| 8                        | 107.853                   | 2.201      | 205.544            | 4.185      | 0.028              |                    |

Table 5 Playback waiting and interruption times (Proposed method)

| Arrival Interval (a (s)) | Playback waiting time (s) | Cumulation | Average (per peer) | Cumulation | Average (per peer) | Average (per piece) |
|--------------------------|---------------------------|------------|--------------------|------------|--------------------|--------------------|
| 2                        | 164.856                   | 3.646      | 46058.436          | 939.406    | 6.266              |                    |
| 4                        | 63.002                    | 1.286      | 150.488            | 3.071      | 0.020              |                    |
| 6                        | 69.036                    | 1.409      | 11.485             | 0.234      | 0.002              |                    |
| 8                        | 53.881                    | 1.100      | 0.000              | 0.000      | 0.000              |                    |

processing time are shown in Table 2. In the simulation, if the number of constraints is between 1 and 50 when the central server calculates the download speed (i.e., $A^p_{ij}$) by solving the linear programming problem, we use the processing time calculated by $y = 133.06e^{0.0118x}$, and if the number of constraints is between 51 and 100, we use the processing time calculated by $y = 336.99e^{0.0048x}$. Additionally, if the processing time calculated by the approximated exponential functions is more than 11404 (µs) (i.e., the maximum processing time in the preliminary experiment), then 11404 (µs) is used as the processing time. In the example of Section 4.4, the processing time is $133.06e^{0.0118×7} (µs)$ because the number of constraints is 7. In the simulation program, the central server solves the linear programming problem by using `lp_solve 5.5.2.0` [21]. When the linear programming problem cannot be solved, $\text{SP}_{j+1} \leftarrow \text{SP}_{j} + \Delta$ (the setting of parameter $\Delta$ is shown in Table 3) is repeated until the linear programming problem can be solved.

We show results of the playback waiting and interruption times under the proposed method and BIS. In BIS, it is difficult to find the optimal combination of the three parameters (i.e., parameters $q$, $p$, and $c$) because BIS has a lot of parameters. In this simulation, the number of combinations of parameters is 48 (i.e., $3 \times 4 \times 4$). In BIS, we first specify a combination of parameters under which the cumulative value of playback interruption times is the minimum value for each average arrival interval (i.e., parameter $a$). As a result, the cumulative value of playback interruption times of BIS is the minimum value for every average arrival interval when parameter $q$ is 3 (Mbyte), parameter $p$ is 0.8, and parameter $c$ is 1. Table 4 shows results of the cumulative value of playback interruption and waiting times under BIS when the above parameters are set. In addition, the average values of playback waiting times per peer (i.e., per a streaming file), and the average values of playback interruption times per a peer (i.e., a streaming file) and per a piece under BIS are shown in Table 4.

On the other hand, the proposed method does not have parameters and parameter tuning is not needed. Table 5 shows results of the cumulative value of playback interruption and waiting times under the proposed method. In addition, the average values of playback interruption times per peer (i.e., per streaming file) and per piece are shown in Table 5. In Tables 4 and 5, the cumulative value of playback waiting times is the sum of periods between the request time for the first piece (piece 1) and its playback starting time for all peers. The cumulative value of playback interruption times is the sum of periods between the playback finishing time of a piece and the starting time of its next piece for all pieces and all peers. Therefore, an average playback waiting time per a peer is calculated by dividing the total number of playback waiting times by the number of all peers (i.e., 49 (peers) excluding peer 0). An average playback interruption time per a peer is calculated by dividing the total number of playback interruption times by the number of all peers (i.e., 49 (peers) excluding peer 0) and an average playback interruption time per a piece is calculated by dividing the total number of playback interruption times by the number of all pieces of all peers (i.e., 150 (pieces) × 49 (peers) = 7350).

From Tables 4 and 5, the proposed method outperforms BIS in terms of the average playback waiting and interruption times, except when the average arrival interval is 2 (s) (i.e., $a = 2$). However, in particular, the average playback interruption time is remarkably high under the proposed method when $a = 2$. In recent years, rapid access to a certain Web server within a short time has increased and this phenomenon is called a flash crowd [22, 23]. There is a possibility that
the flash crowd phenomenon occurs in the cases when peers join the P2P network within very short intervals (e.g., $a = 2$). Therefore, we should reveal the reason why the proposed method does not outperform BIS when $a = 2$, even though the proposed method shows good performance, except when peers join the P2P network within very short intervals (e.g., $a = 2$). Next we describe the reason and an expected solution’s idea for this issue. The reason why the proposed method does not outperform BIS when $a = 2$ is that the total download speed, $\sum_{i\in HP} \sum_{j\in AP} D_{i,j}$, is maximized in objective function (5a) in order to enable the efficient use of the network bandwidth. However, when the maximization of the total download speed is targeted, there is a possibility to not assign excess network bandwidth to peers that hold pieces with tight playback deadlines, even if excess network bandwidth exists. When peers join the P2P network within very short intervals (e.g., $a = 2$), the total requested download speed rapidly increases within a short time and the efficient assignment of excess network bandwidth is needed more in the earlier parts of the streaming file. In order to solve the above issues, we should preferentially assign the excess network bandwidth to peers that hold pieces with tight playback deadlines, in addition to the maximization of the total download speed (i.e., the maximization of the utilization of network bandwidth). However, the proposed method shows good performance in the cases when peers join the P2P network within relatively long intervals (e.g., $a = 4$, $a = 6$, and $a = 8$). In particular, when the average arrival interval is 8 (s) (i.e., $a = 8$), the playback interruption does not occur in the proposed method. We leave the issues of the cases when peers join the P2P network within very short intervals (e.g., $a = 2$) for future work.

### 5.2.1 Analysis of playback interruption

As described in Section 5.2, from Tables 4 and 5, the proposed method outperforms BIS in terms of the average playback waiting and interruption times when the average arrival intervals are 4, 6, and 8 (s) (i.e., $a = 4$, $a = 6$, and $a = 8$). However, the improvement is on the order of milliseconds in terms of the average playback interruption time and the effectiveness of the proposed method is not clear from analyzing the average value. We focus on the cases when the average arrival intervals are 4, 6, and 8 (s) (i.e., $a = 4$, $a = 6$, and $a = 8$) and provide a detailed analysis of the proposed method.

Table 6 shows results of the number of peers that play a streaming file (the streaming file is composed of 150 pieces (piece 1–150)) with one or more playback interruptions in all peers (i.e., peer 1–49), the first half of all peers (i.e., peer 1–25) and the second half of all peers (i.e., peer 26–49) under the proposed method and BIS.

| Arrival Interval (s) | Peer 1–49 (all peers) | Peer 1–25 | Peer 26–49 |
|---------------------|------------------------|-----------|------------|
|                     | Proposed | BIS | Proposed | BIS | Proposed | BIS |
| 4                   | 9        | 38 | 9        | 16 | 0         | 22 |
| 6                   | 5        | 29 | 5        | 12 | 0         | 17 |
| 8                   | 0        | 23 | 0        | 7  | 0         | 16 |

Table 7 Number of playback interruptions

| Arrival Interval (s) | All pieces of peer 1–49 | All pieces of peer 1–25 | All pieces of peer 26–49 |
|---------------------|-------------------------|-------------------------|-------------------------|
|                     | Proposed | BIS | Proposed | BIS | Proposed | BIS |
| 4                   | 22       | 169 |          |     |          | 17  |
| 6                   | 5        | 125 |          | 92 | 0         | 34  |
| 8                   | 0        | 69  |          | 11 | 0         | 58  |

From Table 6, the number of peers that play a streaming file (i.e., piece 1–150) with one or more playback interruptions in peer 1–49, 1–25, and 26–49 under the proposed method is smaller than in BIS when $a = 4$, $a = 6$, and $a = 8$. In particular, peer 1–49 can play a streaming file (i.e., piece 1–150) without playback interruption under the proposed method when $a = 8$. From the fourth and sixth columns of Table 6, in the proposed method, the number of peers that play a streaming file (i.e., piece 1–150) with one or more playback interruptions in peer 26–49 is smaller than peer 1–25. On the other hand, from the fifth and seventh columns of Table 6, the number of peers that play a streaming file (i.e., piece 1–150) with one or more playback interruptions in peer 26–49 is larger than for peer 1–25 in BIS. This is because the proposed method utilizes available bandwidth efficiently while preventing the playback interruption and then efficiently distributes pieces to the whole network from the earlier parts of the streaming file in a short time.

Table 7 shows results of the number of playback interruptions in all pieces of all peers (i.e., piece 1–150) with one or more playback interruptions in all pieces of all peers (i.e., peer 26–49).
peer 1–49), all pieces of the first half of all peers (i.e., piece 1–150 of peer 1–25) and all pieces of the second half of all peers (i.e., piece 1–150 of peer 26–49) under the proposed method and BIS. Here, the number of all pieces of all peers is 7350 (= 150 (pieces) × 49 (peers)), that of all pieces of the first half of all peers is 3750 (= 150 (pieces) × 25 (peers)) and that of all pieces of the second half of all peers is 3600 (= 150 (pieces) × 24 (peers)). The second and third columns of Table 7 show the number of playback interruptions in all pieces of all peers under the proposed method and BIS. Similarly, the fourth and fifth columns show the number of playback interruptions in all pieces of the first half of all peers and the sixth and seventh columns show the number of playback interruptions in all pieces of the second half of all peers.

From Table 7, the number of playback interruptions in all pieces of all peers, all pieces of the first half of all peers and all pieces of the second half of all peers under the proposed method is smaller than BIS when $a = 4, a = 6$ and $a = 8$. From Table 7, the number of playback interruptions in all pieces of all peers under the proposed method is 0 when $a = 8$. As a result, all peers (i.e., peer 1–49) can play all pieces without playback interruption under the proposed method when $a = 8$ in Table 6. On the other hand, when $a = 8$, the number of playback interruptions in all pieces of all peers under BIS is 69 from Table 7, and the number of peers that play a streaming file (i.e., piece 1–150) with one or more playback interruptions under BIS is 23 in Table 6. Therefore, the number of playback interruptions per peer (i.e., per streaming file) under BIS is 3 (= 69 ÷ 23) while that of the proposed method is 0. Similarly, when $a = 4$ and $a = 6$, the number of playback interruptions per peer (i.e., per streaming file) in all peers (i.e., peer 1–49), the first half of all peers (i.e., peer 1–25) and the second half of all peers (i.e., peer 26–49) under the proposed method is smaller than for BIS.

In order to analyze the distribution of playback interruption times, histograms of playback interruption times are shown for $a = 4, a = 6,$ and $a = 8$ in Figs. 10, 11, and 12, respectively.

In [24], the authors report that the playback interruption time is less than 100 (ms) for “sport” content type (i.e., violent (rapid) movement) and is less than 900 (ms) for “talking head” content type (i.e., slight movement) when the MOS (Mean Opinion Score) value is more than 4. It is considered that playback interruption times that are more than 100 (ms) or 900 (ms) have a significant impact on video quality and we use these value (i.e., 100 (ms) and 900 (ms)) as evaluation indexes. From Figs. 10, 11, and 12, in BIS, the frequency gradually decreases as the playback interruption time increases when $a = 4, a = 6,$ and $a = 8$. However, many playback interruption times which are more than 900 (ms) occur when $a = 4, a = 6,$ and
a = 8. The number of interruption times which are more than 900 (ms) is 104 when a = 4, 88 when a = 6, and 48 when a = 8, and that of interruption times which are more than 100 (ms) is 148 when a = 4, 119 when a = 6 and 62 when a = 8. These results have a significant impact on video quality. On the other hand, in the proposed method, several playback interruption times around 6 (s) occur as with BIS when a = 4 from Fig.10, but the playback interruption almost never occurs when a = 6 and a = 8, from Figs.11 and 12. The number of interruption times that are more than 900 (ms) is 25 when a = 4, 3 when a = 6, and 0 when a = 8, and that of playback interruption times that are more than 100 (ms) is 27 when a = 4, 5 when a = 6, and 0 when a = 8. These results have little impact on video quality, compared with BIS.

5.2.2 Analysis of playback waiting time

Similar to the playback interruption, from Tables 4 and 5, the proposed method outperforms BIS in terms of the average playback waiting time when the average arrival intervals are 4, 6, and 8 (s) (i.e., a = 4, a = 6, and a = 8). We focus the cases when the average arrival intervals are 4, 6, and 8 (s) (i.e., a = 4, a = 6, and a = 8).

The playback waiting time means a period between the request time for the first piece (i.e., piece 1) and its playback starting time. The playback waiting time is at least 1 (s) in this simulation because the download time of a piece is at least 1 (s) since the piece size is 1 (Mbyte) and the maximum bandwidth is 8 (Mbps). Table 8 shows the number of peers with the playback waiting time which is more than 1 (s) under the proposed method and BIS. In addition, to analyze distribution of playback waiting times, histogram of playback waiting times are shown when a = 4, a = 6 and a = 8 in Figs.13, 14 and 15, respectively.

Table 8 Number of peers with a playback waiting time that is more than 1 (s)

| Arrival Interval (a (s)) | Proposed | BIS |
|--------------------------|----------|-----|
| 4                        | 3        | 40  |
| 6                        | 9        | 44  |
| 8                        | 6        | 40  |

From Table 8, the number of peers with playback waiting times more than 1 (s) under the proposed method is smaller than under BIS when a = 4, a = 6, and a = 8. In the proposed method and BIS, it is considered that the number of peers with the playback waiting time which is more than 1 (s) is nearly independent of the average arrival intervals. In BIS, the number of peers with the playback waiting time which is more than 1 (s) depends on the selection probability of the first piece (i.e., piece 1). In addition, the selec-
tion probability of the first piece basically depends on the three parameters (i.e., parameters $q$, $p$, and $c$). In this simulation, it is considered that the reason why the number of peers is nearly independent of the average arrival intervals is that the optimal combination of the three parameters is set. That is, the first piece has a certain high priority and the playback waiting time become 1 (s) at a constant ratio under all average arrival intervals (i.e., $a = 4$, $a = 6$, and $a = 8$) On the other hand, in the proposed method, it is considered that the reason why the number of peers is nearly independent of the average arrival intervals is that there is a high possibility that peers requiring the first piece request download speeds higher than other peers requiring the second and subsequent pieces (i.e., peers requiring the first piece have relatively high download speed under all average arrival intervals (i.e., $a = 4$, $a = 6$, and $a = 8$)). However, this needs further theoretical analysis in future work to clarify the reason more.

From Figs.13, 14 and 15, the number of peers with smaller playback waiting times increases according to the increase of the average arrival intervals in BIS because the available bandwidth of each peer increases according to the increase of the average arrival intervals. Almost all the playback waiting times occur within a range of 1 (s) to 5 (s) in BIS. These results have an impact on video quality. On the other hand, from Figs.13, 14 and 15, the distribution of playback waiting times is nearly independent of the average arrival intervals and almost all the playback waiting times occur within a range of 1 (s) to 1.6 (s) in the proposed method because there is a high possibility that peers requiring the first piece request download speeds higher than other peers requiring the second and subsequent pieces. From Figs.13 and 14, two of the playback waiting times occur within a range of 5 (s) to 10 (s) when $a = 4$ and $a = 6$ in the proposed method. From the results of time-series data obtained from the simulation, two of the playback waiting times occurred on both peers 2 and 3. This is because the proposed method assigns excess network bandwidth to a specific peer in the initial stage of the simulation, owing to the maximization of the total download speed. These results may have little impact on video quality because the number of peers with higher playback waiting time is limited to one or two peers.

6. Considerations

In this section, we consider our proposed P2P VoD streaming method from a viewpoint of practical use. In this study, we presented a P2P VoD streaming method to adjust the receiving rate of pieces and showed that it outperformed BIS using computer simulation.

In many previous studies, the behavior of BitTorrent has been investigated from various viewpoints [25]. For example, BitTorrent was evaluated using both computer simulation and a testbed [26] and using theoretical analysis [27]. On the other hand, in BitToS [9], BIS [10], DAW [14] and our proposed method, evaluations were conducted using only computer simulation. Although the effectiveness of the proposed method was demonstrated using computer simulation in Section 5, evaluations using only computer simulation are not enough from a viewpoint of practical use. Therefore, we reveal issues of the proposed method from a viewpoint of practical use. Several important issues exist with respect to how the proposed method actually behaves within real networks. In particular, we focus on environmental factors which are hard to evaluate using only computer simulation. The assumed important issues and the expected solution’s ideas are as follows:

1. How to deal with network bandwidth variations

Similar to BIS [10], we assume that the maximum available bandwidth of the P2P network is limited as $B$ in Section 5. Given the maximum bandwidth ($B$), the assigned download speed ($A$) to satisfy the required download speed ($R$) can be calculated in Section 4. However, there is a possibility that each peer cannot download the selected piece with the assigned download speed ($A$) over a dynamic environment, such as a mobile and wireless network. In particular, if the assigned download speed ($A$) is less than the required download speed ($R$), peers (i.e., users) experience video quality degradation. An expected solution’s idea is that peers switch to downloading or uploading low bitrate video pieces to avoid playback interruptions as described in [28]. However, in this idea, there are multiple bitrate video pieces in the P2P network and it is difficult to manage these multiple bitrate video pieces. To fundamentally solve the problem of a dynamic environment of a mobile network, several mobile VoD streaming methods were proposed in [29] and [30]. On the basis of new knowledge from these studies, we will examine a mobile VoD streaming method in future work.

2. How to deal with an increase of processing time required for solving the linear programming problem

In advance of the simulation experiment, we measure the processing time required for solving the linear programming problem on the specific real computer and then use the exponential functions that are approximated by the measured processing time in the simulation experiment (Section 5). However, when the flash crowed phenomenon occurs, the processing time of the hybrid P2P server is drastically long or the hybrid P2P server may crash as the load increases. Therefore, solutions for flash crowds are needed. An expected solution’s idea is that a router in underlay network or the hybrid P2P server detects flash crowds and then make the number of access to the hybrid P2P server limited (i.e., admission control). In our future work, we will examine new solutions and evaluate them using real
network or testbed assuming bursty request arrivals.

7. Conclusion

In this study, we presented a P2P VoD streaming method which adjusts the receiving rate of pieces, considering the playback deadline of the pieces for reducing the playback interruption time. The proposed method is based on the earliest first strategy with download speed adjustment. The piece selection strategy of the proposed method selects a piece that is the closest to the current playback piece from pieces that have not yet been downloaded. In addition, the proposed method adjusts the download speed of the selected piece to avoid interrupted playback. In the experimental evaluation, we showed that the proposed method outperforms BIS, except when peers join the P2P network within very short intervals (e.g., $a = 2$). In our future work, we plan to propose a solution for assigning excess network bandwidth to peers that hold pieces with tight playback deadlines if excess network bandwidth exists.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 25870696 (Principal Investigator: Masaki Hanada).

References

[1] BitTorrent Inc.: http://www.bittorrent.com, June 2017.
[2] B. Cohen: The BitTorrent protocol specification, http://www.bittorrent.org/beps/bep_0003.html, June 2017.
[3] B. Cohen: Incentives build robustness in BitTorrent, Proc. of 1st Workshop on Economics of Peer-to-Peer Systems, June 2003.
[4] N. Ramzan, H. Park and E. Izquierdo: Video streaming over P2P networks: Challenges and opportunities, Elsevier Journal Signal Processing: Image Communication, Vol. 27, No. 5, pp. 401–411, 2012.
[5] Y. Liu, Y. Guo and C. Liang: A survey on peer-to-peer video streaming systems, Peer to Peer networking and applications, Vol. 1, No. 1, pp. 18–28, Mar 2008.
[6] D. Ghosh, P. Rajan and M. Pandey: P2P-VoD streaming: Design issues and user experience challenges, Advanced Computing, Networking and Informatics, Vol. 2, pp. 169–180, 2014.
[7] Y. Guo, K. Suh, J. Kurose and D. Towsley: P2cast: Peer-to-peer patching scheme for VoD service, Proc. of the 12th international conference on World Wide Web (WWW2003), pp. 301–309, 2003.
[8] T. T. Do, K. A. Hua and M. A. Tantauoui: P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment, Proc. of IEEE International Conference on Communications, Vol. 3, pp. 1467-1472, 2004.
[9] A. Vlavianos, M. Iliofotou and M. Faloutsos: BiToS: Enhancing BitTorrent for supporting streaming applications, Proc. of IEEE INFOCOM 2006, Vol. 1, pp. 1–6, 2006.
[10] S. Sakashita, T. Yoshihisa, T. Haru and S. Nishio: A method to reduce interruption time in P2P streaming environments, Journal of Information Processing of Japan, Vol. 53, No. 3, pp. 1045–1054, 2011 (in Japanese).
[11] C. Dana, D. Li, D. Harrison and C-N. Chuah: BASS: BitTorrent assisted streaming system for video-on-demand, Proc. of IEEE MMSP 2005, pp. 1–4, Oct. 2005.
[12] D. Erman, K. De Vogeleer and A. Popescu: On piece selection for streaming BitTorrent, Proc. of Swedish National Computer Networking Workshop (SNCNW2008), April 2008.
[13] D. Erman: On BitTorrent media distribution, PhD thesis, Blekinge Institute of Technology, March. 2008.
[14] P. Sandvik and M. Neovius: The distance-availability weighted piece selection method for BitTorrent: A BitTorrent piece selection method for on-demand streaming, Proc. of Advances in P2P Systems (AP2PS 2009), pp. 198–202, October 2009.
[15] Y. Huang, T. Z. J. Fu, D. M. Chiu, J. C. S. Liu and C. Huang: Challenges, design and analysis of a large-scale P2P-VoD System, Proc. of the ACM SIGCOMM 2008, pp. 375–388, 2008.
[16] M. Fouda, T. Taleb, M. Guizani, Y. Nemoto and N. Kato: On supporting P2P-based VoD services over mesh overlay networks, Proc. of the 28th IEEE Global Telecommunications Conference (GLOBECOM2009), pp. 3928–3933, 2009.
[17] Y. Zhou, D. M. Chiu and J. C. S. Liu: A simple model for chunk-scheduling strategies in P2P streaming, IEEE/ACM Transactions on Networking, Vol. 19, No. 1, pp. 42–54, 2011.
[18] V. Pai, K. Tanilman, V. Sambamurthy, K. Kumar and A. B. Mohr: Chainsaw: Eliminating trees from overlay multicast, Proc. of 4th International Workshop Peer-to-Peer Systems (IPTPS2005), pp. 127–140, 2005.
[19] X. Zhang, J. Liu, B. Li and Y. Yum: Coldstream/DONet: A data-driven overlay network for efficient live media streaming, Proc. of IEEE INFOCOM 2005, 2005.
[20] N. Magharei and R. Rejaie: PRIME: Peer-to-peer receiver-driven mesh-based streaming, IEEE/ACM Transactions on Networking, Vol. 17, No. 4, pp. 1052–1065, 2009.
[21] Introduction to lp_solve 5.5.2.0: http://web.mit.edu/lpsolve/doc/, June 2017.
[22] G. Banga and P. Druschel: Measuring the capacity of a Web server under realistic loads, World Wide Web, Vol. 2, No. 1–2, pp. 69–83, June 1999.
[23] A. Stavrou, D. Rubenstein and S. Sahu: A lightweight, robust P2P system to handle flash crowds, IEEE Journal on Selected Areas in Communications, Vol. 22, No. 1, pp. 6–17, 2006.
[24] S. van Kester, T. Xiao, R. E Kooij, K. Brunstrom and O. K. Ahmed: Estimating the impact of single and multiple freezes on video quality, Proc. of SPIE, Vol. 7865, December 2011.
Masaki Hanada received his B.E. degree in Resources Engineering from Waseda University in 1996, and M.S. and D.S. degrees in Global Information and Telecommunication Studies from Waseda University, Japan in 2003 and 2007, respectively. He is an Associate Professor in the Department of Information Systems, Tokyo University of Information Sciences. His research interests include traffic control and resource management in communication networks. He is a member of the RISP, IEEE and IEICE.

Hidehiro Kanemitsu received his B.S. degree in Science from Waseda University, Japan in 2002, and M.S. and D.S. degrees in Global Information and Telecommunication Studies from Waseda University, Japan in 2006 and 2012, respectively. His research interests include parallel and distributed computing, grid, peer-to-peer computing, and web service technology. He is currently an Assistant Professor at the Global Education Center, Waseda University, Japan. He is a member of the RISP, IEEE and IEICE.

(Received July 24, 2017; revised November 15, 2017)