Reducing Interruption Time by Segmented Streaming Data-Scheduling in Hybrid Broadcasting Environments

Tomoki Yoshihisa

Cybermedia Center, Osaka University, Japan
yoshihisa@cmc.osaka-u.ac.jp

Abstract - In hybrid broadcasting environments, clients play streaming data such as video or audio while receiving them. Playback interruptions occur when data reception is later than the start time of data play. Although methods of reducing interruption time have been proposed, these methods have large drawbacks in that the server often broadcasts data that most clients have already received. Hence, I propose an interruption-time reduction method in which the server schedules some segmented streaming data. By broadcasting segments according to a schedule, the server can deliver the segments that many clients have not yet received, thus effectively reducing the interruption time.

Keywords: Broadcast Schedule, Continuous Media, Interruption Time, Video-on-Demand

1 INTRODUCTION

There has recently been a great deal of interest in hybrid broadcasting environments and in streaming delivery using these environments. In this type of delivery, servers deliver streaming data such as video or audio from both broadcasting systems and communication systems. In broadcasting systems (e.g., TV and radio), servers broadcast data according to predetermined broadcast schedules and can deliver data to many clients concurrently. In communication systems (e.g., the Internet), clients can receive the data they want by requesting them directly from servers at arbitrary timings. The hybrid broadcasting environments of these systems are effective for streaming delivery, because clients can receive data from both broadcasting systems and communication systems.

In streaming delivery in hybrid broadcasting environments, clients play streaming data while receiving them from both systems. When data reception is later than the start time of data play, playback interruptions occur. A short interruption time is preferable for viewers to enjoy video or audio. Methods of reducing the interruption time for streaming delivery in hybrid broadcasting environments have been proposed [1]–[6]. Here, *interruption time* means the elapsed time during which playback interruptions occur. This includes the time elapsed from when a request is made to play the data to when the data start to play.

In previous methods, the data are divided into segments of fixed sizes and the server broadcasts the data segment that is

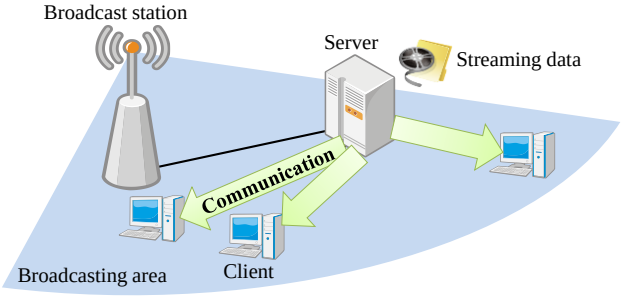

Figure 1: A hybrid broadcasting environment

requested by the client that has the shortest *margin time* (i.e., the margin between the current time and the time when the next interruption occurs). However, this approach has the following drawback. The newest client is the one with the shortest margin time, and this client requests the first segment. The server therefore often broadcasts this segment but many clients have already received it. (For details see Section 3.2.) The server can therefore reduce the interruption time effectively by broadcasting segments that the majority of clients have not yet received.

Here, I propose an interruption time reduction method for hybrid broadcasting environments[1]. In the proposed method, the server can broadcast segments that many clients have not yet received by scheduling some segments dynamically. This is the key point of the proposed method and is different from conventional methods used in hybrid broadcasting environments. Appropriate number of segments scheduled can be estimated by computer simulations. The system can find this by simulating interruption times changing the parameters. By broadcasting segments according to the created schedule, the server does not always broadcast only the first segment, even when new clients request data play. By broadcasting segments that many clients have not yet received, the server can reduce the average interruption times when there are large numbers of clients.

I also describe how to determine the number of scheduled segments in this paper. By adjusting the estimated interruption time so that it is close to the appropriate value, the proposed method can effectively reduce the average interruption time. I evaluate the proposed method under some average request arrival intervals and reveal that the method can reduce average interruption times further than can conventional methods.

[1] This research was supported in part by the Strategic Information and Communications R&D Promotion Programme (SCOPE) of the Ministry of Internal Affairs and Communications, Grant-in-Aid for Scientific Research (B) number 15H02702, and Grant-in-Aid for Challenging Exploratory Research number 26540045.

The rest of the paper is organized as follows. Section 2 explains related work. The proposed methods are presented in Section 3 and evaluated in Section 4. Finally, I present my conclusions in Section 5.

2 RELATED WORK

Methods of reducing interruption times have been proposed before [7]–[12]. After explaining hybrid broadcasting environments, I will introduce some of the methods used for streaming delivery with reduced interruption times in hybrid broadcasting environments.

2.1 Hybrid Broadcasting Environments

Figure 1 shows the hybrid broadcasting environment assumed in this paper. The clients in the broadcasting area can receive data from the broadcasting system. Also, they can ask the server for the data they want and receive them from the communication system. The broadcast station delivers the data via broadcast channels and is managed by the server. The server has streaming data and can broadcast the data to the clients by using the broadcast station. It can also send the data to the clients by unicasting using the communication system. The streaming data consist of segments. The segments are units for playing the streaming data, such as GOPs (Groups of Pictures) for MPEG-encoded streaming data. Examples of streaming delivery in hybrid broadcasting environments are the delivery of video data to TVs or delivery to smart phones connected to the Internet. In this case, the broadcasting system is a terrestrial broadcasting system or a satellite broadcasting system and the communication system is the Internet.

2.2 Methods for Interruption Time Reduction

In the UVoD (Unified Video-on-Demand) method [2], the server broadcasts the streaming data cyclically via each broadcast channel. Because the time to the start of each broadcast cycle is delayed for all broadcast channels, clients get more opportunities to receive the data. When an interruption is about to occur, the client tries to receive the data that causes the interruption directly from the server via the communication system.

In the SSVoD (Super-Scalar Video-on-Demand) method [3], the server broadcasts the data in the same way as with UVoD. However, unlike with UVoD, the server does not send the requested data to the clients until other clients request the same data. After the server has received a number of requests, the server multicasts the requested data to the clients.

In the NBB VoD (Neighbors-Buffering Based Video-on-Demand) method [4], the server broadcasts the data in the same way as with UVoD, but it then uses a peer-to-peer (P2P) approach. Clients receive the desired data from other clients that have already received them. If the other clients do not have the data, the server sends them directly to the client requesting them.

In the above methods, the server broadcasts all of the data repeatedly, although clients can receive some parts of the data from the communication system. In contrast, FC (First segment from Communication), MC-LB (Middle segment from Communication and Last segment from Broadcast), and MC-LC (Middle segment from Communication and Last segment from Communication) methods have been proposed [5]. In these methods, the server predicts the data that the clients will receive from the communication system and eliminates the predicted data from the broadcast schedule. These three methods differ in terms of which data are eliminated. However, the broadcast schedule is static and the methods do not consider the data that the clients have already received.

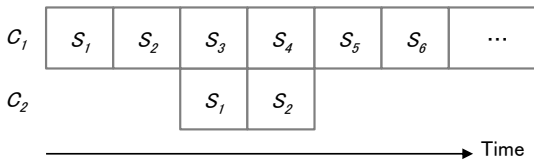

Figure 2: Broadcast schedule using the DHB method

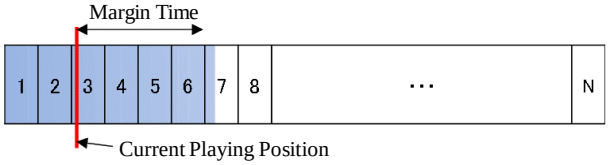

Figure 3: Example of margin time

In the DHB (Dynamic Heuristic Broadcasting [7]) method, the server dynamically broadcasts the requested data by using other broadcast channels. Figure 2 shows a broadcast schedule under the DHB method. C_i (i = 1, 2, ...) denotes the broadcast channels. The data are divided into six segments, S_1, ... , S_6. The server broadcasts S_1, ... , S_6 sequentially via C_1. As an example, suppose that a new client requests data play during broadcasting S_2. The client can receive S_3, ... , S_6 from C_1. To make the interruption time for this client short, the server broadcasts S_1 and S_2 via C_2. However, this method considers data reception only from the broadcasting system, not from the communication system.

In the SET-C (Shortest Extra Time per Client) method [6], the server determines the data to broadcast dynamically, taking into account margin time. As explained in Section 1, the margin time is the time between the current time and the time when the next interruption will occur, and it is calculated from the data that the clients have already received. Figure 3 gives an example of margin time. The vertical red line indicates the current time and the colored area indicates the data that the client has already received. At the time shown in the figure, the client is receiving the 7th segment and the margin time is the time until that segment will start to play. The SET-C method can reduce interruption time, because the probability that the client will avoid interruptions is increased by broadcasting the segment that the client with the shortest margin time requests. In the SET-C method, however, the server broadcasts the first segment when a new client requests data play, because the server determines the next segment to broadcast every time a broadcasting block finishes. Accordingly, the server often broadcasts segments that many other clients have already received. This is the main problem of the SET-C method.

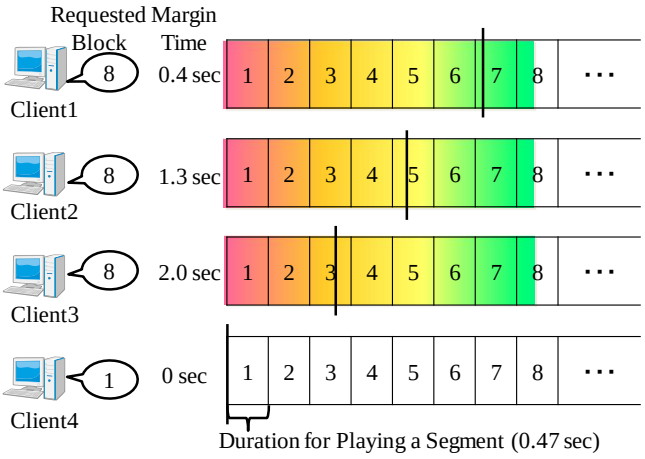

Figure 4: A figure to explain the problem

3 PROPOSED METHOD

This section explains the interruption-time reduction method that I propose. After explaining the assumed system environments, I explain the data delivery for broadcasting systems and for communication systems.

3.1 Assumed System Environments

This research assumes streaming delivery on hybrid broadcasting environments (explained in Section 2.1). Because the server has many streaming data and it is difficult to predict which data the clients will play, the clients do not receive data before they request data play. Clients play streaming data from the beginning to the end continuously, without fast-forwarding or rewinding. Their storage capacity is larger than the size of the requested streaming data. The broadcast station uses one broadcast channel to broadcast one data stream, as occurs in actual systems.

3.2 Main Problem of the Existing (SET-C) Method

In previously proposed methods, the advantage of the broadcasting system (i.e., concurrent delivery of the same data to multiple clients) is the most apparent when the number of clients receiving the same data increases. However, as explained in Sections 1 and 2, the server broadcasts the first segment when new clients request data play, because it is the new clients that have the shortest margin time. Here, the term *new client* means a client that has just started data play from the beginning of the streaming data. Accordingly, the server often broadcasts segments (including the first segment) that many clients have already received. Figure 4 gives an example. Clients 1 to 3 have received the preceding seven segments. The black vertical lines indicate the playing positions of each client. In this situation, the clients are requesting data play at different times and the playing positions differ among the clients, although the data received are the same. Here, suppose the case in which the new client, Client 4, requests data play. Clients 1 to 3 are receiving segment 8 and their margin times are respectively 1.5, 4.8, and 9.2 s. The margin time of Client 4 is 0 s, because this client has not received any segments. In this case, under the previously proposed (SET-C) method, the server broadcasts segment 1, because the margin time of Client 4 is the shortest and Client 4 has requested segment 1. However, the other clients have already received segment 1 and the server cannot exploit the advantage of the broadcasting system. One of the solutions is for the server not to broadcast the segment that the new client requests, but instead to broadcast the segment that many clients have not yet received.

3.3 G-SET-C (Grouped SET-C) Method

The proposed G-SET-C method increases the probability that the server will broadcast the segment that many clients have not yet received, thus reducing interruption time. The server broadcasts the segment that the new client requests after it has broadcast some of the other segments.

The G-SET-C method does not directly consider the number of clients requesting the same segments. The reason is that direct consideration by the server of the number of clients requesting the same segments gives a longer interruption time than the G-SET-C method, as shown in the evaluation results (see the G-MRB method in Section 4.2). This is because clients receive their requested segments from the communication system while the server broadcasts the scheduled segments. Thus, the advantage of the broadcasting system (delivery of data to all clients concurrently) does not operate. However, with the G-SET-C method, by scheduling some of the segments, the server can broadcast segments that many clients have not yet received. This is the key point of the G-SET-C method. The details are given in the next section.

3.3.1 Effectiveness of Grouped Scheduling

The G-SET-C method uses grouped scheduling. That is, the server schedules some segments every time broadcasting of scheduled segments finishes. This avoids the problem described in Section 3.2.

There are two reasons why grouped scheduling achieves the broadcasting of segments that many clients have not yet received. The first is that the server does not consider the margin times of new clients until the next scheduling time. The server can therefore broadcast segments other than the first one, even when a new client comes along. Because many clients have already received the first segment, the server can avoid broadcasting that segment and can instead broadcast segments that most clients have not yet received. In the original SET-C method, the server soon broadcasts the first segment when a new client comes along, because the server schedules only one segment at the finish of every broadcasting segment.

The second reason is that the segment that each client does not have and that is the closest to the current playing position for each client gradually becomes the same as time proceeds. This is because the segment requested by the client with the shortest margin time is not broadcast for a long time, and the broadcast segments are eventually received by all clients. This phenomenon (i.e., in which clients *catch up* with other clients that have started playing the data earlier) is also

observed with the original SET-C method. However, for the first reason given above, clients easily catch up with other clients with the G-SET-C method. For these reasons, with the G-SET-C method, the server can broadcast segments that many clients have not yet received without considering the number of clients requesting the same segments. The main difference between the G-SET-C method and the SET-C method is the number of scheduled segments.

3.3.2 Delivery on Broadcasting Systems

With the G-SET-C method, the server schedules G segments when creating the next broadcast schedule. The server creates broadcast schedules every time the broadcasting of all scheduled segments finishes. The server creates the schedules by considering the segments requested by clients that will have shorter margin times. For this, the server predicts the margin times of all clients.

Let $S(g)$ denote the time to start broadcasting the gth segment (g = 1, ... , G) included in the broadcast schedule, and let $E_i(g)$ denote the predicted margin time of client i. The server can get the actual margin time (=E_i (1)) because the server creates the broadcast schedule at $S(1)$ and this is the current time. Let $R_i(g)$ denote the time for client i to finish receiving the segment requested at $S(g)$. The time needed to broadcast one segment is B_b, and the duration of play of one segment is P_b.

First, when $S(f+1) < R_i(f)$ (f = 1, ... , G-1)—that is, client i cannot finish receiving the requested segment before the broadcasting start time of the f + 1th scheduled segment—the margin time at $S(f+1)$ decreases by the amount of time taken to play one segment. So, $E_i(f+1) = E_i(f) - B_b$. If this is a negative value, $E_i(f+1) = 0$. Next, suppose the case when $R_i(f) < S(f+1)$, that is, client i can finish receiving the requested segment before the broadcasting start time of the f + 1th scheduled segment. When an interruption occurs before the finish time of reception ($E_i(f) < R_i(f) - S(f)$), the client restarts playing the data after its reception. So, $E_i(f+1) = R_i(f) + P_b - S(f+1)$. This always takes a positive value, because $B_b < P_b$ in this research. Otherwise, if ($R_i(f) - S(f) < E_i(f)$), the margin time increases by the amount of time it takes to play one segment. So, $E_i(f+1) = E_i(f) - B_b + P_b$. Hence,

$$E_i(f+1) = \begin{cases} 0 & (S(f+1) < R_i(f), E_i(f) < B_b) \\ E_i(f) - B_b & (S(f+1) < R_i(f), E_i(f) > B_b) \\ R_i(f) + P_b - S(f+1) & (S(f+1) > R_i(f), E_i(f) < R_i(f) - S(f)) \\ E_i(f) - B_b + P_b & (S(f+1) > R_i(f), E_i(f) > R_i(f) - S(f)) \end{cases}$$

In the proposed G-SET-C method, the server schedules the segment that is requested by the client j that satisfies the following equation for each g. N is the set of clients. When some clients have equivalent margin times, the server schedules the segment that was requested by the client that requested the initial data play earliest.

$$E_j(g) = \min_{i \in N} E_i(g)$$

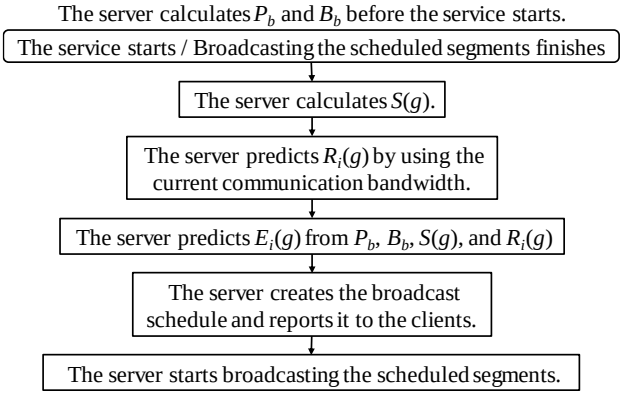

Figure 5: Flow of the G-SET-C method

Figure 5 shows the flow of the G-SET-C method. The server can calculate P_b and B_b before the streaming data delivery service starts, because these values are constant during the service. When the service starts, or when the server finishes broadcasting the scheduled segments, the server starts the process of broadcast schedule creation. First, the server calculates $S(g)$. The value can be calculated from the current time and B_b. After that, the server predicts $R_i(g)$ by using the current communication bandwidth. The server also predicts $E_i(g)$. From the predicted values of $E_i(g)$, the server creates broadcast schedules. The server reports the broadcast schedule to the clients for their determination of the segment to receive from the communication system. The server then starts broadcasting the scheduled segments.

3.3.3 Margin Time Prediction

With the G-SET-C method, to calculate the predicted margin time $E_i(g)$ (g = 1, ... , G), the server needs to predict the time taken to finish receiving segment $D_i(g)$ that client i requests at $S(g)$.

When the client receives $D_i(g)$ from the broadcasting system, the server can calculate $R_i(g)$ by using $S(g)$, because the server can calculate the time to start broadcasting each scheduled segment. For example, if $D_i(g)$ is scheduled to the eth segment in the broadcast schedule, $R_i(g) = S(e) + B_b$.

When the client receives $D_i(g)$ from the communication system, the server predicts $R_i(g)$ by using the communication bandwidth for client i at the time of creation of the broadcast schedule, C_i. C_i is the bandwidth between the server and client i. First, client i may have received part of $D_i(1)$ at that time. So, $R_i(1)$ is given by the remaining data size divided by C_i. Next, for $D_i(f+1)$ (f = 1, ... , G–1), $R_i(f+1) = R_i(f)$ if $D_i(f+1)$ is the same segment as $D_i(f)$. Otherwise, $R_i(f+1)$ is the value obtained by adding the segment size divided by C_i to $R_i(f)$.

The following time sequence shows how the server decides on broadcast segments in the G-SET-C method.

1. The server predicts the margin times. For prediction of the time taken for each client to finish receiving segments from the communication system, the server uses the current bandwidths.
2. The server creates the broadcast schedule on the basis of the predicted margin times.
3. The server notifies all clients of the broadcast schedule.

4. When a client finishes receiving segments from the communication system, the client decides on the next segment to receive from the communication system. For this decision, clients use the notified broadcast schedule.
5. The client starts receiving the decided segment from the communication system. If the client's bandwidth changes greatly, the chosen segment will differ from the server's prediction. Otherwise, it will be the same.

The predicted margin times, including R_i (g), are based on the bandwidths for communication with the client at the time of creation of the broadcast schedules. This is merely a prediction. So, margin times can differ greatly from actual times if the bandwidths change greatly. Otherwise, they are close to the actual times. The sequence therefore does not include cyclic dependency.

3.3.4 Determination of G

The interruption time depends on the number of scheduled segments, G. As confirmed in the evaluation section, the interruption time is reduced further by giving an appropriate value for G. The appropriate value depends on the average arrival interval, the bit rate, etc. and is difficult to estimate. However, the system can give a value close to the appropriate one by performing a computer simulation of the average interruption time. Some network simulators have been developed and we can use these simulators. The system can adjust the value to make it close to the appropriate value while delivering the data.

For example, service providing systems can find the most appropriate value of G by modifying the parameters for simulations based on actual values. They can get actual values by measuring them for a period, e.g. day, week, month, after stopping streaming data delivery services. Simulation results in Section 4 are measured using the author developed simulator. By using such a simulator, the systems can find the value of G that is close to the appropriate value.

3.3.5 Example of Broadcast Schedule Creation

Here, I give an example of the creation of a broadcast schedule by using a simulation result. The simulated situation is shown in Table 1. In this situation, the time to play a segment P_b = 0.469 s and the time to broadcast a segment B_b = 0.125 s. At 1982.899 s after the beginning of the simulation, client 395 is playing segment 22 and the segment that the client does not have and that is the closest to the current playing position is segment 23. The actual margin time is the time between the current time and the time to start playing segment 23 and is 0.445 s.

In this example, G = 2. The server schedules segment 23 as the first scheduled segment since E_i (1) = 0 (i = 394, 396, 397, 398, 399). When the predicted margin times are equivalent, the server schedules the segment requested by the client that requested data play earliest. Here, again, E_i (g) (g = 1, ... , G) is the predicted margin time at the time to start broadcasting the gth scheduled segment. Next, the server predicts the margin times of all clients at the time of the start of broadcasting of the second scheduled segment so as to determine the second segment to be included in the broadcast schedule. The margin times of clients 394 and 395 are equal

Table 1: Example of segment scheduling

Time [s]	Client ID	Playing segment	Closest Non-received segment	Margin time [s] (=E_i (1))
1982.899	394	22	23	0
	395	22	23	0.445
	396	2	3	0
	397	1	2	0
	398	1	2	0
	399	0	1	0
1984.149	394	24	25	0.250
	395	24	25	0.250
	396	5	7	1.000
	397	3	4	0
	398	3	4	0.125
	399	2	4	0.750
1986.649	394	29	40	5.375
	395	29	30	0.500
	396	8	10	0.750
	397	6	10	1.875
	398	6	10	1.875
	399	6	10	1.875

to P_b minus B_b, because they request segment 23. So, at 1982.899 s, E_{394} (2) = E_{394} (1) – B_b + P_b = 0 – 0.125 + 0.469 = 0.344 s and E_{395} (2) = E_{395} (1) – B_b + P_b = 0.445 – 0.125 + 0.469 = 0.789 s. In this simulation, the time for client 396 to finish receiving segment 3 from the communication system R_{396} (1) = 1982.993 s. So, the margin time E_{396} (2) = R_{396} (1) + P_b – S(2) = 1982.993 + 0.469 – (1982.899 + 0.125) = 0.438 s. The times for other clients to finish receiving their requested segments are later than S(2) and E_j (2) = 0 (j = 397, 398, 399). Therefore, the server schedules segment 2, which is requested by client 397, as the second scheduled segment.

Just at 1986.649 s, the server again schedules two segments and first schedules segment 30 as the first scheduled segment, since E_{395} (1) is the shortest and client 395 requests the segment. Next, the server predicts the margin times for each client and decides on the second scheduled segment. At 1986.649 s, the time to start broadcasting the second segment is 1986.649 + 0.125 = 1986.774 s. The predicted margin times for this time are; E_{394} (2) = E_{394} (1) – B_b = 0.5375 – 0.125 = 5.25 s, E_{395} (2) = E_{395} (1) – B_b + P_b = 0.500 – 0.125 + 0.469 = 0.844 s, E_{396} (2) = E_{396} (1) – B_b = 0.750 – 0.125 = 0.625 s, and E_{397} (2) = E_{398} (2) = E_{399} (2) = 1.875 – 0.125 = 1.75 s. These values are the predicted margin times at 1986.649 s and are different from those at 1982.899 s through the symbols are the same. The client that has the shortest predicted margin time is client 396 and the client requests segment 10. So the server schedules segments 30 and 10 at 1986.649 s.

The catch-up phenomenon explained in Section 3.3.1 is observed in the following way. The time of 1982.899 s is immediately after client 399 requests data play, at which time the client requests segment 1. At 1984.149 s, the segments that client 399 has received are the same as those received by clients 397 and 398, and they are requesting segment 4. That is, client 399 catches up with clients 397 and 398. Also, at 1986.649 s, the segments that have been received by clients 396 to 399 are the same, and they are requesting segment 10.

Table 2: Simulation parameter values

Item	Value
Duration of data streaming	25 min
Bit rate	2 Mbps
Broadcast bandwidth	8 Mbps
Clients' communication bandwidth	1 Mbps
Server's communication bandwidth	30 Mbps
Segment size	125.012 Kbytes
Header size	12 bytes

Clients 397 to 399 catch up to client 396. The advantage of the broadcasting systems (i.e., that the server can deliver the same data to all clients) operates well here, because the segment that clients 396 to 399 do not have, and that is the closest to the current playing position, is the same at 1986.649 s.

3.3.6 Delivery via Communication Systems

As in most of the previously proposed methods, clients start receiving blocks from the communication system when they request data play. Clients receive the block that satisfies the following conditions:

- The block will cause interruptions if the client waits for its broadcasting.
- The block can be received faster than reception from the broadcasting system.
- The block is closest to the current playing position.

If there are no blocks that satisfy these conditions, to avoid redundant communication the clients do not receive the blocks from the communication system. They request the next block when they finish receiving each block.

4 EVALUATION

In this section I present the results of simulations used to evaluate the proposed G-SET-C method.

4.1 Evaluation Environments

Table 2 shows the evaluation parameter values. The assumed streaming data are MPEG2-encoded (2 Mbps) movie data with a duration of 25 min. The segments consist of GOPs and the data size is the same as the general GOP data size (0.5 s). The broadcast bandwidth is 8 Mbps, assuming that the broadcasting system is a terrestrial one. The assumed communication system is the Internet. The communication bandwidths for all clients are equivalent. If the total communication bandwidth for the clients exceeds the server's communication bandwidth, the server's communication bandwidth is apportioned equally to each client. The header includes information on the identifiers for the streaming data and segments and the number of segments. The data size for each item of information is 4 bytes and the header size becomes 4 × 3 = 12 bytes. G indicates the number of scheduled segments.

In the simulation, clients request data play when they arrive in the system and leave the system when they finish playing the data. I measured interruption times until the number of clients arriving reached 4000. Interruption times saturate when the number of clients arriving is 4000 and this was a sufficient number of clients to calculate average interruption times.

4.2 Comparison Methods

The performance of the original SET-C method is equivalent to that of the G-SET-C method when $G = 1$. Other comparison methods are explained below.

- BCD-BE-AHB (Broadcast- and Communication-based Delivery-BE-AHB) Method

  This method applies the BE-AHB method proposed in [12] to hybrid broadcast environments. Data delivery on the broadcasting system is similar to the original. The data is divided into some segments. The data sizes for segments are calculated from broadcasting bandwidths. The segments are repeatedly broadcast via each channel. Data delivery on the communication system under this method uses the same algorithm as that under the proposed method. The broadcast schedule is static with this method, whereas that under the proposed method is dynamic.
- G-MRB (Grouped-Most Requested Block) Method

  With this method, the server schedules the top G segments requested by the majority of clients. When there are some segments for which the number of clients requesting the segment is the same, the server schedules the segment that is requested by the client that started data play earliest. When the number of segments requested is less than G, the server broadcasts all requested segments.
- G-LTIT-C (Grouped-Longest Total Interruption Time per Client) Method

  With this method, the server schedules the segments that are requested by the client with the longest interruption time at the time of broadcast of each scheduled segment. Let $I_i(g)$ denote the predicted interruption time for client i at the time of broadcast of the gth scheduled segment (g = 1, ... , G), i.e., $S(g)$. The server can get $I_i(g)$ by asking each client their interruption times. $I_i(f+1)$ (f = 1, ... , G–1) is given by the following equation.

$$I_i(f+1) = \begin{cases} I_i(f) + B_b - E_i(f) & (S(f+1) < R_i(f), E_i(f) < B_b) \\ E_i(f) & (S(f+1) < R_i(f), E_i(f) > B_b) \\ I_i(f) + R_i(f) - S(f) - E_i(f) & (S(f+1) > R_i(f), E_i(f) < R_i(f) - S(f)) \\ I_i(f) & (S(f+1) > R_i(f), E_i(f) > R_i(f) - S(f)) \end{cases}$$

  In the G-LTIT-C method, the server schedules the segment that is requested by the client j that satisfies the following equation for each g. When some clients have the same interruption time, the server schedules the segment that was requested by the client that started data play earliest.

$$I_j(g) = \max_{i \in N} I_i(g)$$

  The G-LITT-C method differs from the SET-C method and the G-SET-C method, even when $G = 1$, because the G-LITT-C method considers interruption time. The SET-C

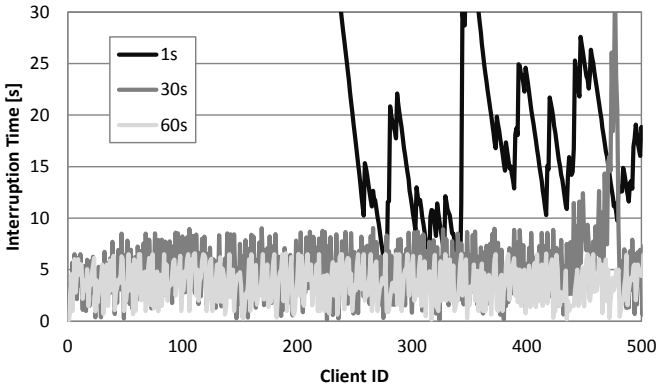

Figure 6: Clients' interruption times

method and the G-SET-C method consider margin time. Moreover, in the G-LITTC method the server schedules several segments, whereas in the SET-C method it schedules only one segment.

4.3 Interruption Time

The simulated interruption times for each client under the proposed G-SET-C method are shown in Fig. 6. The figure shows only the interruption times for the preceding 500 clients. The simulated average arrival intervals are 1, 30, or 60 s. The horizontal axis is the client ID, which is given along with the request time for data play, and the vertical axis is the interruption time. We can see that the interruption time has an upper limit, though it has some dispersion. Hence, the average interruption time is used as an evaluation criterion.

4.4 Influence of the Number of Scheduled Segments

The time taken to create the broadcast schedule depends on the number of scheduled segments, *G*. The probability that segment 1, which is requested by new clients, will be broadcast increases as the interval taken to create the broadcast schedule shortens, and the server often broadcasts segment 1. Hence, I measure the average interruption times under different *G* values. Because the characteristics of the results change with the average arrival interval, the following sections discuss each case individually.

4.4.1 Cases When the Average Arrival Interval is 1 s

Figure 7 shows the average interruption times when the average request arrival interval is 1 s. Figure 8 is an enlargement of Fig. 7.

From these figures, we can see that in most cases the proposed G-SET-C method gives the shortest average interruption time. This is because the server does not always broadcast the segment requested by new clients but instead schedules the segments requested by other clients. Thus, the probability of broadcasting a segment that many clients have not yet received increases. Discussions of each method follow.

With the G-SET-C method, the average interruption time decreases as *G* increases when *G* is less than 90. In such cases, as *G* increases, the server broadcasts more segments that are

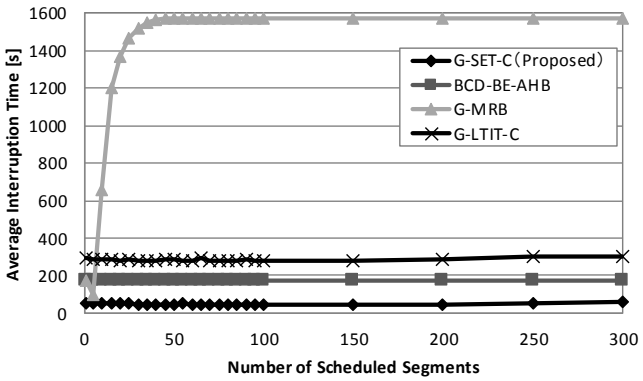

Figure 7: Average interruption time (average request arrival interval 1 s)

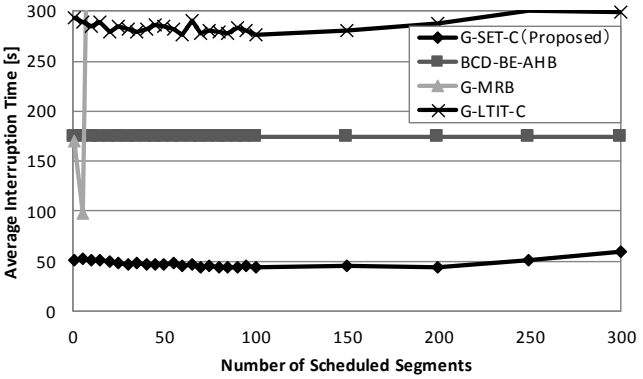

Figure 8: Details of average interruption time (average request arrival interval 1 s)

requested by clients other than new clients, that have just started data play. Therefore, the chances that the server will broadcast segments that many clients have not yet received increases and the average interruption time is reduced effectively. When *G* is greater than 90, the average interruption time increases as *G* increases. In such cases, the time elapsed until the server broadcasts segment 1, which is requested by new clients, increases too much and the average interruption time lengthens. When the value of *G* is much larger, the average interruption time does not change greatly, because new clients receive segment 1 from the communication system while the server broadcasts other segments.

The average interruption time under the G-SET-C method approaches that under the BCD-BE-AHB method when *G* is too large. The reason is that the server schedules the segment that was requested by the client that started playing the data earliest. The broadcast schedules under the proposed method therefore become similar to those under the BCD-BE-AHB method and the average interruption time becomes equivalent when *G* is too large.

The BCD-BE-AHB method does not create the broadcast schedule dynamically, and the broadcast schedule does not depend on *G*. So, the method is not effective and the average interruption time is longer than that under the proposed G-SET-C method.

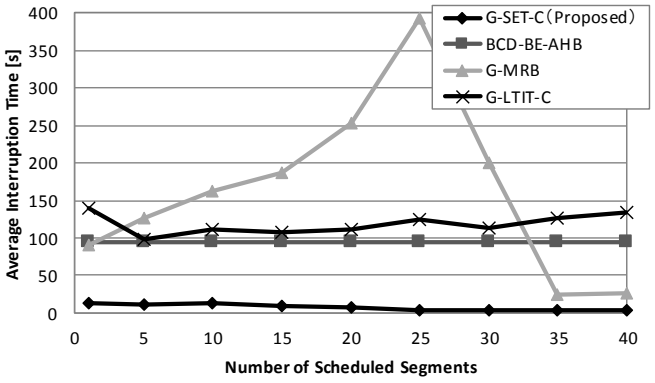

Figure 9: Average interruption time (average request arrival interval 30 s)

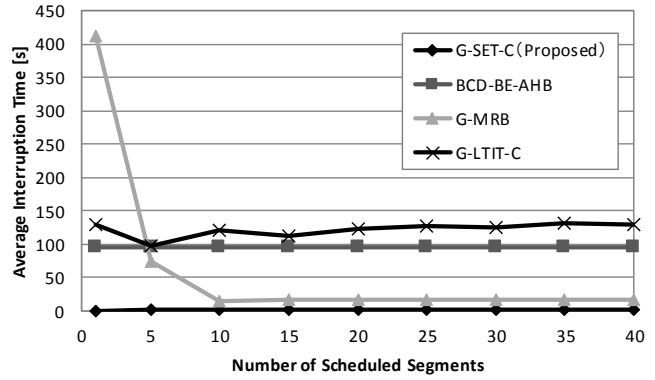

Figure 10: Average interruption time (average request arrival interval 60 s)

With the G-MRB method, the average interruption time is shortest when G is 2. When G is 1 or 2, segments that are requested by clients with long interruption times are included in the broadcast schedule. However, when G is larger than 2, the interval for creating the broadcast schedule lengthens. Therefore, the probability that such segments are included in the broadcast schedule decreases and the interruption time increases. Also, when G is larger than 50, the average interruption time is constant. This is because the number of clients that request segments is less than G and the broadcast schedule does not change even if G increases.

With the G-LTIT-C method, the average interruption time is shortest when G is 85. This is because the server broadcasts more segments requested by clients other than new clients. However, the time elapsed until the server broadcasts the first segment increases as G increases. For the same reason, the value of G that gives the shortest average interruption time is the same as that with the G-SET-C method.

4.4.2 Cases When the Average Arrival Interval is 30 s

Figure 9 shows the average interruption times when the average request arrival interval is 30 s. The proposed G-SET-C method gives the shortest average interruption time in all cases. This is because, for the same reason as when the average arrival interval is 1 s, the server schedules a number of segments and does not always broadcast the segment that the new client requests.

Compared with the result when the average arrival interval is 1 s, the average interruption time under the G-MRB method changes greatly. This is because the probability that several clients request the same segment decreases as the average request arrival interval lengthens. The influence of the number of clients that request the same segment is larger with the G-MRB method than with other methods because the G-MRB method considers the number of segments requested directly.

When G is less than 25, the average interruption time increases as G increases. This is because the interval for creating broadcast schedules lengthens as G increases. However, when G is larger than 25, the interruption time decreases as G increases. This is because the server can broadcast many segments requested by clients at the time of creation of the broadcast schedule.

4.4.3 Cases When the Average Arrival Interval is 60 s

Figure 10 shows the average interruption times when the average request arrival interval is 60 s. In this case the proposed G-SET-C method also gives the shortest average interruption time.

Compared with the results for the other average arrival intervals, one of the main differences is that the average interruption time under the G-MRB method decreases as G increases when G is small. This is because clients can receive their requested segment from the communication system while the server broadcasts the scheduled segments. Because the server can broadcast more segments requested at the time of creation of the broadcast schedule, the average interruption time decreases as G increases. When G is larger than 10, the average interruption time is constant, because the number of segments requested is less than G. Therefore, in these cases the server actually does not schedule G segments.

5 CONCLUSION

In this paper, I proposed a segmented streaming data scheduling method for streaming delivery in hybrid broadcasting environments. Different from conventional methods, in the proposed G-SET-C method, the server schedules some segments considering the margin time until the next interruption. This approach leads to the broadcasting of segments that many clients have not yet received. The G-SET-C method thus contributes to further reduce interruption time. I also described how to determine the appropriate number of scheduled segments in this paper. My evaluation revealed that in many cases the proposed method could reduce the average interruption time further than with conventional methods.

In the future, I am planning to propose a method that considers stopping data play and applies the P2P data delivery techniques.

REFERENCES

[1] V. Gopalakrishnan, B. Bhattacharjee, K. Ramakrishnan, R. Jana, and M.K. Vernon, "CPM: Adaptive Video-on-Demand with Cooperative Peer Assists and Multicast," IEEE INFOCOM 2009, pp. 91-99 (2009).

[2] J.Y.B. Lee, "UVoD: An Unified Architecture for Video-on-Demand Services," IEEE Communication Letters, Vol. 3, No. 9, pp. 277-279 (1999).

[3] J.Y.B. Lee and C.H. Lee, "Design, Performance Analysis, and Implementation of a Super-Scalar Video-on-Demand System," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, Issue 11, pp. 983-997 (2002).

[4] T. Taleb, N. Kato, and Y. Nemoto, "Neighbors-Buffering-based Video-on-Demand Architecture," Signal Processing: Image Communication, Vol. 18, Issue 7, pp. 515-526 (2003).

[5] M. Umezawa, T. Yoshihisa, T. Hara, and S. Nishio, "Interruption Time Reduction Methods by Predicting Data Reception for Streaming Delivery on Hybrid Broadcasting Environments," Proc. IEEE Pacific Rim Conference Communications, Computers and Signal Processing, pp. 185-190 (2011).

[6] S.W. Carter, J.F. Paris, S. Mohan, and D.D.E. Long, "A Dynamic Heuristic Broadcasting Protocol for Video-on-Demand," Proc. IEEE International Conference on Distributed Computing Systems, pp. 657-664 (2001).

[7] T. Yoshihisa, "Dynamic Data Broadcasting Methods for Streaming Delivery on Hybrid Broadcasting Environments," Proc. International Workshop on Advances in Data Engineering and Mobile Computing, pp. 470-475 (2015).

[8] D.L. Eager and M.K. Vernon, "Dynamic Skyscraper Broadcast for Video-on-Demand," Proc. of International Workshop on Advances in Multimedia Systems, pp. 18-32 (1998).

[9] H. Kim and H.Y. Yeom, "Dynamic Scheme Transition Adaptable to Variable Video Popularity in a Digital Broadcast Network," IEEE Transactions on Multimedia, Vol. 11, No. 3, pp. 486-493 (2009).

[10] J.B. Kwon. and H.Y. Yeom, "Adjustable Broadcast Protocol for Large-scale Near Video-on-Demand Systems," Computer Communications, Vol. 28, No. 11, pp. 1303-1316 (2005).

[11] Q. Zhang and J.F. Paris, "A Channel-based Heuristic Distribution Protocol for Video-on-Demand," Proc. IEEE International Conference on Multimedia and Expo, Vol. 1, pp. 245-248 (2002).

[12] T. Yoshihisa and S. Nishio, "A Division-based Broadcasting Method Considering Channel Bandwidths for NVoD Services," IEEE Transactions on Broadcasting, Vol.59, Issue 1, pp. 62-71 (2013).

(Received September 26, 2015)
(Revised March 10, 2016)

Tomoki Yoshihisa received his Bachelor's, Master's, and Doctor's degrees from Osaka University, Osaka, Japan, in 2002, 2003, 2005, respectively. Since 2005 to 2007, he was an assistant professor at Kyoto University. In January 2008, he joined Cybermedia Center, Osaka University as a senior lecturer and in March 2009, he became an associate professor. From April 2008 to August 2008, he was a visiting researcher at University of California, Irvine. His research interests include video-on-demand, broadcasting systems, and webcasts.