Implementation and Evaluation of Division-based Broadcasting System for Webcast

Yusuke Gotoh[1], Akihiro Kimura[1]
[1]Graduate School of Natural Science and Technology,
Okayama University, Kita, Okayama 700-8530, JAPAN
gotoh@cs.okayama-u.ac.jp, en19616@s.okayama-u.ac.jp

JDIM
Journal of Digital Information Management

ABSTRACT: *Due to the recent popularization of IP multicasts, the continuous broadcasting of audio or video media data is attracting great attention. Although servers can concurrently deliver data to many clients, they have to wait until their data are broadcast. In division-based broadcasting, many researches have proposed scheduling methods to reduce waiting time. A division-based broadcasting system has also been proposed to evaluate these schemes. However, this system does not consider several problems that are assumed in actual networks. In this paper, we implement a division-based broadcasting system for webcasts. In our proposed system, we evaluate the effectiveness of reducing both the waiting and interruption times using conventional methods.*

Subject Categories and Descriptors: H.3.4 [Systems and Software]; H.3.4 [Information networks]

General Terms: Design, Management, Measurement

Keywords: Continuous Media Data, Division-based Broadcasting System, Scheduling Method, Waiting Time

Received: 2 March 2015, Revised 12 April 2015, Accepted 19 April 2015

1. Introduction

Due to the recent popularization of IP multicasts, such continuous media data broadcasting as audio or video are attracted great attention. For delivering streaming data for webcasts, clients select programs and watch them. Since clients need to wait after making their selection before they can play, they become impatient and want to watch without interruption.

In IP networks, there are mainly two types of delivery systems: Video on Demand (VoD) and broadcasting systems. In VoD systems, the server start delivering data sequentially based on client requests. However, the available bandwidth increases in proportion to the number of clients, and waiting times under VoD systems increase. On the other hand, in broadcasting systems, the server repeatedly delivers the same data to many clients at a constantly available bandwidth. When clients receive data sequentially, waiting occurs between requesting and receiving the initial part of the data. In webcasts, many clients require many types of data from servers, and the size of each data becomes large. Therefore, in this paper, we design and implement a broadcasting system for webcasts.

Many studies employ the division-based broadcasting technique, which reduces waiting time by dividing data into several segments and frequently broadcasting the precedent segments. These scheduling methods make a broadcast schedule that considers the situation in actual network environments. However, since most scheduling methods in broadcasting schemes evaluate the waiting times in simulation environments, we need to evaluate them using scheduling methods in network environments.

1.1 Motivation

We previously proposed a division-based broadcasting system [1] that evaluated scheduling methods in network environments. However, conventional system do not consider several problems that are assumed in actual networks. These problems increase the waiting times before playing the data and the interruption times while playing them. In this paper, we implement and evaluate a

division-based broadcasting system for webcasts. In our proposed system, we evaluate the effectiveness of reducing interruption time using conventional methods. The main idea of our paper is that we propose a scheme to solve such problems as additional information that increases waiting times before playing data as well as interruption times while playing them.

The remainder of the paper is organized as follows. The division-based broadcasting system is explained in Section 2, and related works are introduced in Section 3. We explain the problems with conventional division-based broadcasting systems in Section 4. Our proposed system is designed in Section 5. We implement and evaluate our system in Sections 6 and 7. Finally, we conclude our paper in Section 8.

2. Division-based Broadcasting System

2.1 Waiting Times in VoD and Broadcasting Systems

In webcasts, there are mainly two types of delivery systems: VoD and broadcasting. First, we calculate waiting times for VoD and broadcasting systems. Next, we explain waiting times for broadcasting data that are divided into several segments.

The situations that cause waiting times in both systems are shown in Figure 1. In the VoD system, the server starts delivering data sequentially based on client requests. Waiting times under VoD systems are roughly equal to receiving times. When the server repetitively broadcasts continuous media data, clients have to wait until the first portion of the data is broadcast.

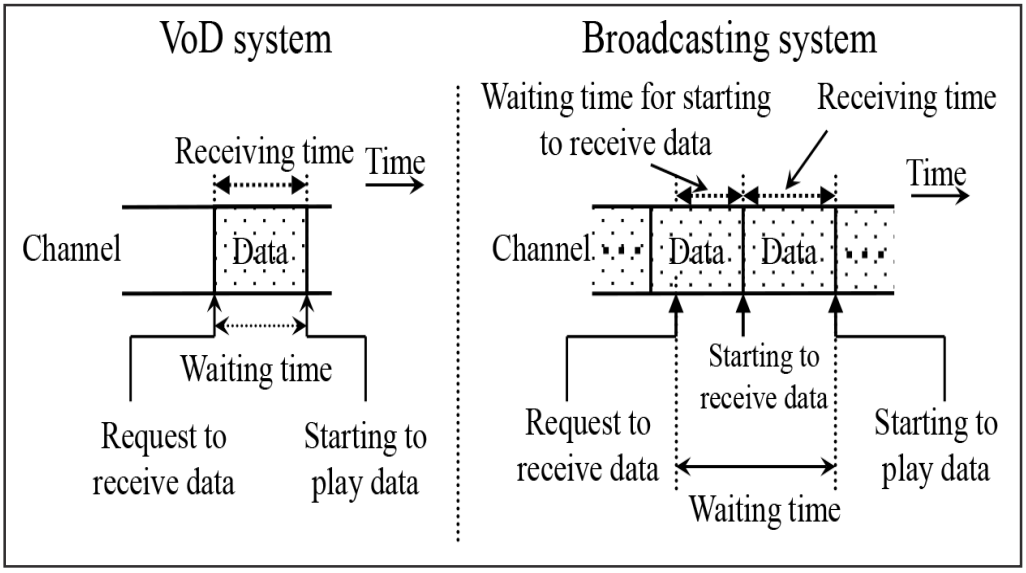

Figure 1. Waiting times in VoD and broadcasting systems

For example, suppose a server broadcasts MPEG2-encoded music clip data using a 15 Mbps broadcasting system. The consumption rate of the data is 5.0 Mbps, and the playing time is seven min. The data size becomes 5.0 × 7 × 60 / 8 = 262.5 Mbytes. In a simple method, since the server broadcasts the data repetitively without dividing them, the broadcasting time is 7 × 60 × 5.0 / 15 = 140 sec. The minimum waiting time is given by the client that starts receiving the data just before the start of the broadcast cycle. The minimum waiting time is 140 sec., which equals the time necessary to broadcast the data. The maximum waiting time is given by the client that starts receiving the data just after the start of the broadcast cycle. Since the client has to wait until the first portion of the data is broadcast in the next cycle, the maximum waiting time is twice the broadcast cycle: $140 \times 2 = 280$ sec. Since the waiting time between the maximum and minimum waiting times is uniformly distributed, the average waiting time is $(280 + 140) / 2 = 210$ sec. To reduce the waiting time, many methods employ a division-based broadcasting technique, which reduces it by dividing the data into several segments and frequently broadcasting precedent segments.

2.2 Scheduling Methods to Reduce Waiting Times

In division-based broadcasting, since the data are divided into several segments, it is important to schedule segments without interrupting the continuous play of clients. An example of the division-based broadcasting technique using the Fast Broadcasting (FB) [2] method is shown in Figure 2. Here, the broadcast bandwidth is divided into several channels. The bandwidth for each channel is equivalent to the consumption rate. In this case, the server uses three channels, the data are divided into three segments, S_1, S_2, and S_3.When the total playing time is seven min., the playing time of S_1 is calculated to be one min., S_2 is two min., and S_3 is four min.The consumption rate is 5.0 Mbps. The bandwidth for each channel is the same as the data consumption rate.The server repetitively broadcasts S_i ($i = 1, 2, 3$) by broadcast channel C_i. The available bandwidths for clients are 15 Mbps. Clients can store broadcasted segments in their buffers while playing the data and can play each segment after receiving it. In this case, when clients finish playing S_1, they finish receiving S_2 and can play S_2 continuously. When finish playing S_2, they finish receiving

S_3 and can play S_3 continuously. In this case, the waiting time is the same as the time needed to receive only S_1 and the average waiting time is one min. since clients can receive the broadcasted segments from their midstream.

An example of division-based broadcasting using the Bandwidth Equivalent-Asynchronous Harmonic Broadcasting (BE-AHB) method [3] is shown in Figure 3. The server divides the data into two segments, S_1 and S_2, and the playing time of the data is 60 sec. The consumption rate is 0.9 Mbps, and the bandwidth for each channel is 1.5 Mbps. The server repetitively broadcasts S_i (i=1, 2) by broadcast channel C_i. In this case, the minimum waiting time is 12.7 sec., which is the same as the time needed to receive only S_1.

3. Related Works

3.1Scheduling Methods in Division-based Broadcasting

In division-based broadcasting systems, scheduling methods have been proposed to reduce waiting times. The Fast Broadcasting (FB) method [2] sets the ratio of the data size for each segment and schedules it without interruption. When the ratio of the data size in the first segment is 1, the FB method sets it in the n th segment as 2^{n-1}. The server schedules segments using channels where all the available bandwidth are equal. When the available bandwidth of each channel equals the consumption rate, the waiting time under

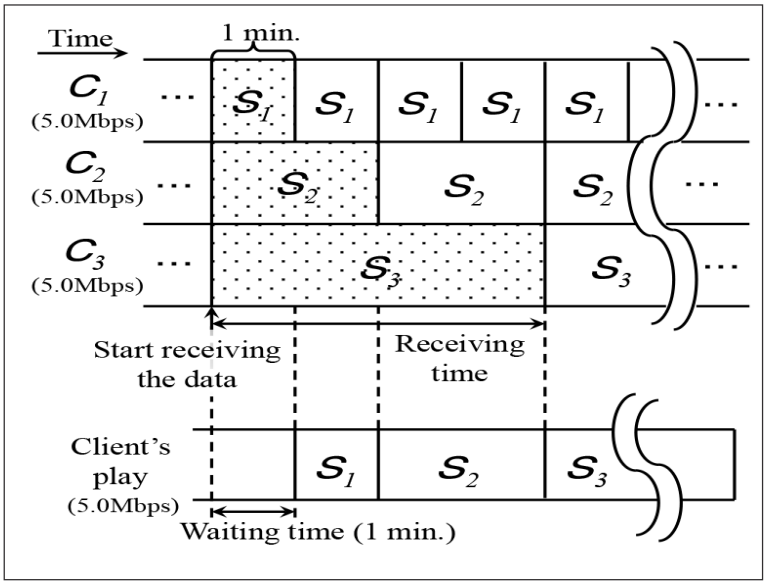

Figure 2. Example of broadcasting situation with interruptions

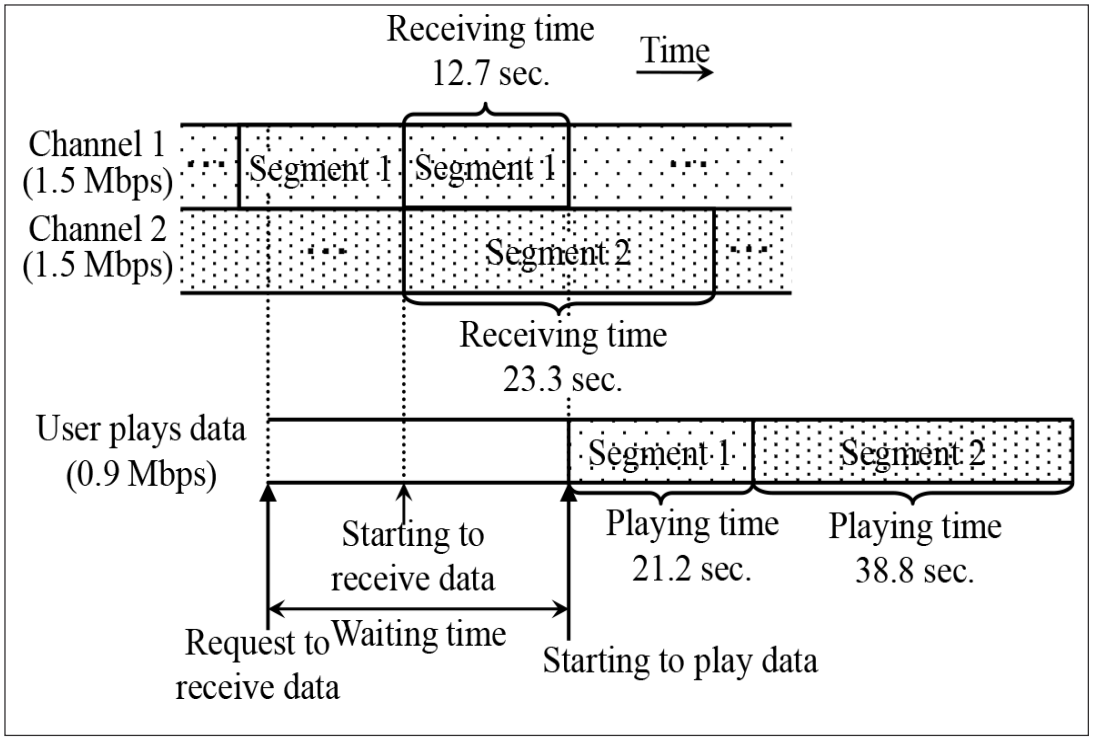

Figure 3. Example of broadcasting schedule in BE-AHB method

the FB method can be reduced more than that under the BE-AHB method [3]. However, when the available bandwidth of each channel is less than the consumption rate, interruption occurs while playing the data.

In Optimized Periodic Broadcast (OPB) [4], each bit of data is separated into two parts. The server uses several broadcast channels and distributes each segment on each channel. When clients completely receive the precedent parts of the content, they start receiving the remaining portions of the data. Since clients can get the sub-segment data in advance, waiting times can be reduced. However, the bandwidth increases as the amount of contents increases.

In Heterogeneous Receiver-Oriented Broadcasting (HeRO) [5], the server divides the data into K segments. Let J be the data size for the first segment. The data sizes for the segments are $J, 2J, 2^2J, \ldots, 2^{k-1}J$.

The server broadcasts these segments using K channels. However, since the data size of the K_{th} channel becomes half of the data, clients may have waits and interruptions.

The Generalized Fibonacci Broadcasting (GFB) method [6] makes a broadcast schedule using the Fibonacci sequence. When the data size ratios in the first and second segments are 1 and 2, the GFB method sets the ratio of the data size in the n^{th} segment $(n \geq 3)$ to $(n-2)+(n-1)$ and creates schedules for clients with several sizes of available bandwidth. Clients with small available bandwidth can reduce their waiting times.

The Catch and Rest (CAR) method [7] combines scheduling using the replicated channels in HeRO with scheduling that divides segments in the GFB method. Clients who have different bandwidth and buffer sizes use the CAR method. Clients with small buffer size and available bandwidth can reduce their waiting times by considering cases without receiving segments.

In Layered Internet Video Engineering (LIVE) [8], clients feedback virtual congestion information to servers to support both bandwidth sharing and transient loss protection. LIVE minimizes the total distortion of all the participating video streams and maximizes their overall quality. Also, several researches have studied reliable broadcasting in wireless networks with packet loss probability [9, 10]. In our paper, we evaluate a division-based broadcasting system that considers the receiving opportunity.

Other scheduling methods focus on the receiving performance of clients [11, 12] and the variable bit rate (VBR) [13, 14]. When a server delivers data to clients using bandwidth that is sufficiently larger than the consumption rate who play it by buffering, they can reduce their waiting times. However, clients must have high performance memory and a processor. We need to evaluate a system in a situation where clients with various computer performances can simultaneously use division-based broadcasting with scheduling methods.

3.2 d-Cast

As explained in Section 3.1, many scheduling methods have been proposed to reduce waiting times in division-based broadcasting. However, most scheduling methods calculate waiting times by computational simulations. To evaluate the scheduling method in actual networks, a division-based broadcasting system called d-Cast [1] has been proposed.

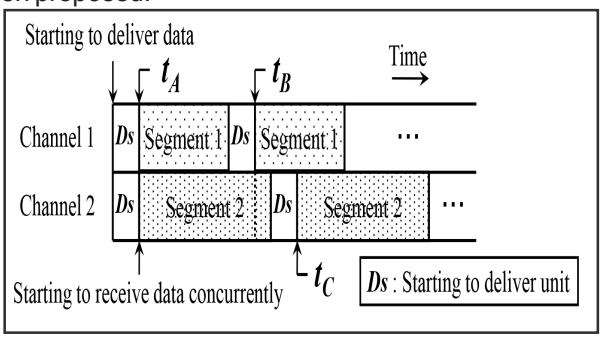

Figure 4. Setting for receiving data in d-Cast

The setting for receiving data in d-Cast is shown in Figure 4. The server delivers data called the start-delimiter header with the movie data. The start-delimiter header is information which is sent to clients about starting to deliver data.

After receiving the start-delimiter, clients start receiving segments and can play the data after completely receiving the initial segment. In d-Cast, clients cannot start receiving data from the middle of segments.

4. Problems with Conventional Division-based Broadcasting System

Since the scheduling methods described in Section 3 suppose a situation where processing load and packet loss are not considered, they fail to consider the effect on division-based broadcasting systems. To evaluate scheduling methods in actual networks, we proposed a division-based broadcasting system called d-Cast [1], which introduces several types of conventional broadcasting methods and designs internet broadcasting systems based on a network environment with a server and clients.

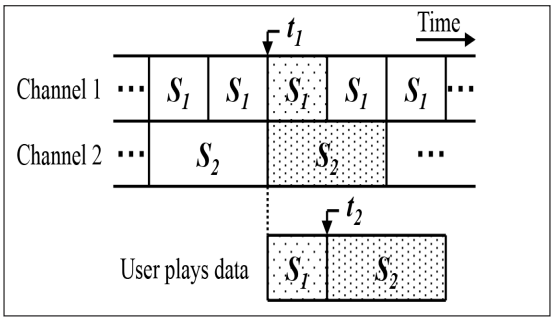

Figure 5. Broadcast schedule under FB method without start delimiter

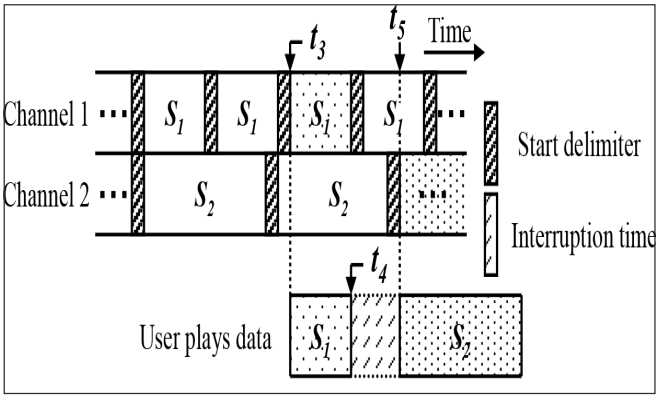

Figure 6. Broadcast schedule under FB method with start delimiter

However, d-Cast has three problems: the effect of broadcasting schedules by the start delimiter header, the synchronization of timing for delivering data, and sequential playback. In Sections 4.1, 4.2, and 4.3, we explain the details of these problems in d-Cast.

4.1 Effect of Broadcasting Schedule by Start Delimiter Header

d-Cast ignores the effect of delivering the header of the start-delimiter. When it uses a scheduling method that needs to synchronize the timing to start delivering segments for several channels, interruptions occur. To explain the mechanism that causes interruptions, an example of division-based broadcasting is shown in Figures 5 and 6. The server separates the data into one part S_1 with two parts S_2 and broadcasts them repeatedly using channels S_1 and S_2.

In Figure 5, the client who requires data at t_1 can start S_1 and S_2 concurrently. Clients can store the buffer of S_2 to play S_1. When the client finishes playing S_1 at t_2, it can play S_2 using the stored buffer. Therefore, we need to set the same starting time to deliver S_1 and S_2 to play data without interruption.

Figure 6 considers the start-delimiter header. In this case, the starting times to play S_1 and S_2 are not the same. For example, when the client requires data at t_3, it can only start receiving S_1. Therefore, interruption time occurs between t_4 when the client finishes playing S_1 and t_5 when it starts playing S_2.

4.2 Synchronization of Timing for Delivering Data

As explained in Section 4.1, we need to address scheduling that includes the start delimiter in actual networks. To solve this problem, there are two schemes. One sets the dividing ratio based on the start delimiter. In this scheme, we need to precisely identify the start delimiter. However, since there are several types of start delimiters based on communication protocols, this is difficult.

The other scheme synchronizes the starting times of delivering the data. In it, the server waits to start delivering the next segment until it completely delivers all the segments. In addition, the server does not need to precisely identify any start-delimiter header. Therefore, the effectiveness of this scheme is lower than a scheme that sets the dividing ratio based on the start-delimiterheader. However, the start delimiter is only added to S_1. In addition, the data size in the start delimiter is less than 1,500 bytes, which isthe data size of the Ethernet frame. When the server delivers other segments than S_1,it does not decrease the delivering efficiency.

4.3 Sequential Playback

In this paper, we design and implement a function of sequential playback using HTTP, which does not need private servers and clients. In contrast to download playback, the sequential playback function can play data by receiving the minimum data size. Although the function of the sequential playback is already designed for such streaming systems as YouTube, conventional division-based broadcasting systems have not designed it. In division-based broadcasting, our proposed system has a unique communication protocol that can be for both multicasts and broadcasts.

5. Design

5.1 Design Policy

To solve the d-Cast problem, we improve the setting for receiving data. By adding the start delimiter to only the initial segment, clients can start receiving data regardless which segments they receive. The design policy in our proposed system is shown in Figure 7.Clients start receiving segments for all the channels after completely receiving the start delimiter of S_1. For example, when the client starts receiving S_1 after completely receiving the start delimiter in S_1 at t_A, the server delivers S_2 in C_2. In this case, clients start receiving the middle of S_2 and receive its total amount.

In addition, the data format of the designed broadcasting system is shown in Figure 8. There are two types of formats: start-delimiter and information-delimiter headers. The start-delimiter header is added and only delivered in initial segment. Clients can start receiving data after receiving the start-delimiter header.

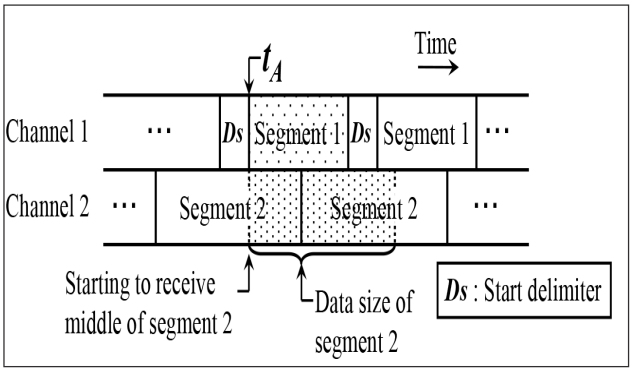

Figure 7. Setting data in our proposed system

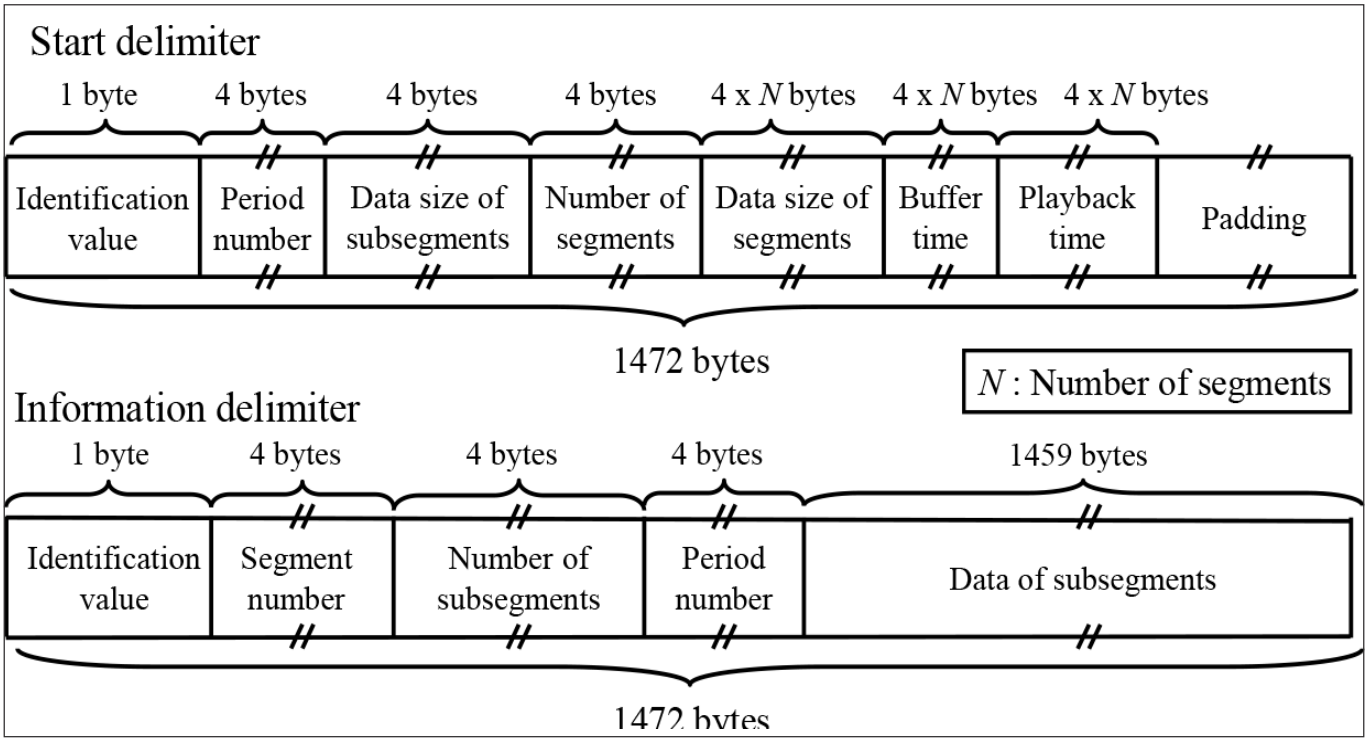

Figure 8. Data Format

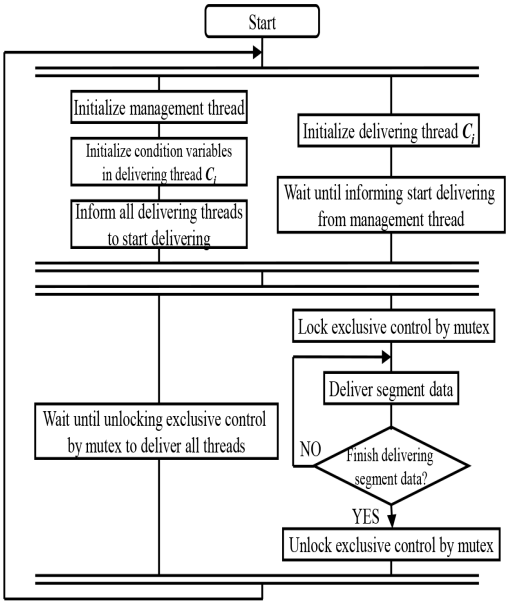

Figure 9. Flowchart of server

5.2 Algorithm

The algorithm for designing our proposed system is below. Based on it, we designed the server and clients.

(1) The server broadcasts data repeatedly using the scheduling method.

(2) When the client requests data, it connects the session with the server and waits until it receives the start delimiter.

(3) After receiving the start delimiter, the client starts receiving segments and stores them in its buffer.

(4) After completely receiving all segments, the client plays the data.

(5) When the client finishes playing the data, it disconnects the session from the server.

5.3 Server Design

Actual networks have many types of communication protocols. In our paper, we design and implement a scheme to synchronize the timing of delivering data. In this scheme, the delivery processing of each channel and the timing for delivering the data are performed in parallel. In the mechanism, the server sends start and stop messages in parallel processing that it synchronizes using exclusion control. The processing flow of the server in our proposed scheme is shown in Figure 9. The number of channels is N. The server uses broadcast channels C_i ($i = 1, \ldots, N$),

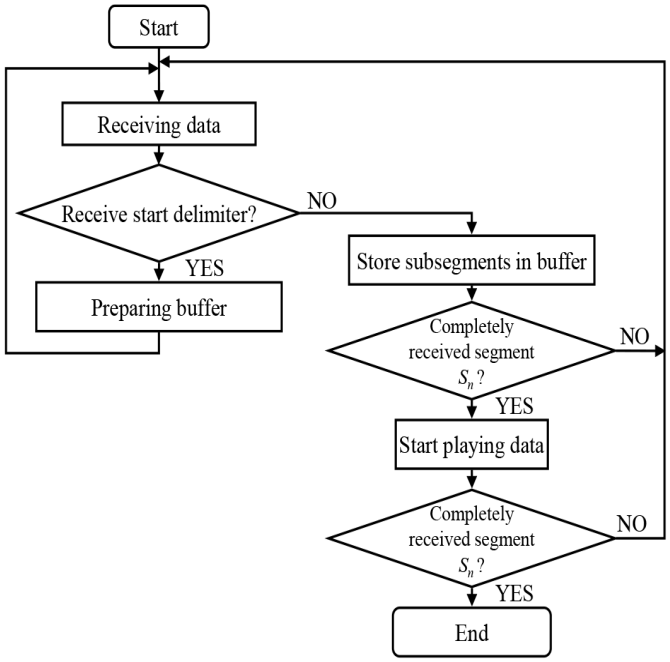

Figure 10. Flowchart of Client

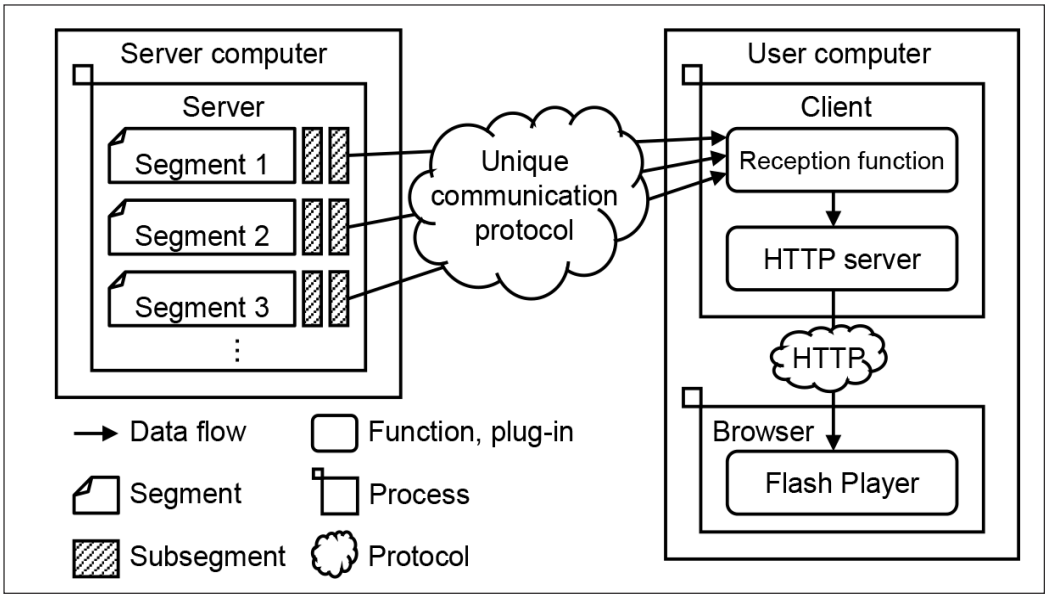

Figure 11. Configuration of our Proposed system

sets the processing to manage the timings of the delivering data as managing and delivering threads, and parallelizes them.

The managing thread initializes the variables to send the starting and stopping messages of the parallelized processes called a condition variable. Next, the managing thread sends a starting message to deliver the data to all the threads using the condition variable. After sending the message, the exclusion control is released by a mechanism called mutex that manages it. Finally, the managing thread waits for the execution until the process is synchronized among all the delivering threads.

The delivering thread of C_i waits to be executed until the message to start delivering the data is sent from the managing thread. After sending the message from the manag ing thread, the delivering thread sets exclusion control for the mutex and delivers segments using *Ci*. After the segment has been delivered, the delivering thread releases the exclusion control for mutex and waits until the message is sent to restart delivering the segments using a condition variable.

5.4 Client Design

The flowchart of the clients is shown in Figure 10.First, they check whether their received data include a start-delimiter header. If they received a start-delimiter header, they acquire the data size of each segment, the waiting time to prevent interruptions, and its playing time. If they did not receive a start-delimiter header, the clients discard the data and receive the next segment of data. When clients receive an information delimiter header, they store information about the numbers of segments and sub seg

ments in their buffer. They repeat this process until the total amount of the received data equals the data size of the segment or they receive a sub segment that has already been received. When clients completely receive all the segments, they can watch the movie data by a media player.

5.5 Sequential Playback

In this paper, we design and implement a sequential playback function using HTTP, which does not need private servers and clients. The configuration of our proposed system is shown in Figure 11. First, the server divides the data into several segments. Next, it divides each segment into several sub segments and delivers them using several channels. After receiving a HTTP request from the browser, the client starts receiving subsegments using the receiving function. In the receiving function, sub segments are sorted and are sent to the HTTP server. Finally, the HTTP server delivers the sub segments to the browser using HTTP.

6. Implementation

6.1 Development Environment

Based on the solution explained in Section 5, we improved our division-based broadcasting system. Its hardware configuration is shown in Figure 12. In our assumed hardware configuration, one server sends data to N clients by an IP network. Our assumed system environment is summarized below.

- The broadcast channel has upper bandwidth.
- The communication protocol uses UDP/IP.
- The server sets the scheduling method and divides the data before starting to deliver them.
- The server repeatedly delivers data.
- Clients can start playing the data when they receive their beginning segment.
- Both server and client computers have C compilers.

6.2 System Setting

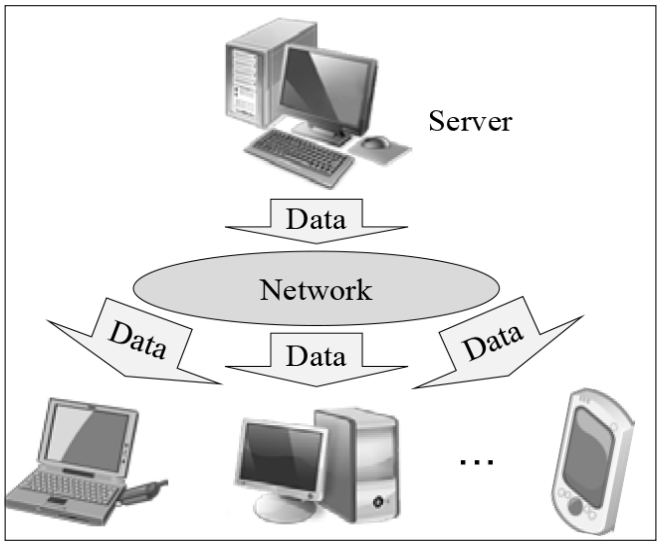

Figure 12. Hardware Configuration

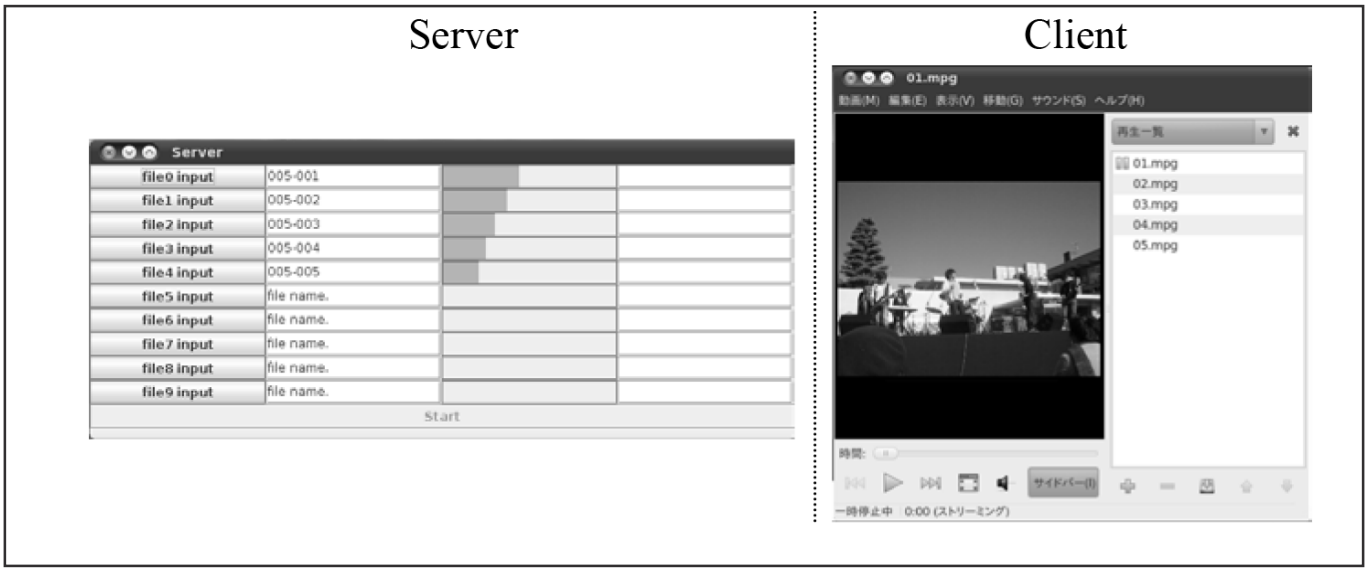

Figure 13. Screen Shot

Server	CPU	Intel® Core™ 2 Duo CPU E7500@2.93GHz
	Memory	2.0GB
	OS	Ubuntu 10.04
	NIC	RTL8101E/RTL8102E PCI Express Fast Ethernet controller

Dummynet	CPU	Intel® Core™ 2 Duo CPU E7500@2.93GHz
	Memory	2.0GB
	OS	FreeBSD 8.1
	NIC_1	RTL8101E/RTL8102E PCI Express Fast Ethernet controller
	NIC_2	RTL8169SC
	NIC_3	RTL8169SC

Client	CPU	Intel® Core™ 2 Duo CPU E7500@2.93GHz
	Memory	2.0GB
	OS	Ubuntu 10.04
	NIC	RTL8101E/RTL8102E PCI Express Fast Ethernet controller

Table 1. Measurement Environment

A screenshot of our system is shown in Figure 13. We designed our proposed system using C programming language. Our proposed system can operate on several types of operating systems including Windows. In addition, we implemented a function with which servers can operate the process by a Graphical User Interface (GUI) to easily set the delivery of several segments.

7. Evaluation

7.1 Evaluation Environment

We constructed an environment that evaluates our proposed system on an actual network. The computer performance in our evaluation is shown in Table 1. We established an UDP/IP connection with a server and a client by Gigabit Ethernet. In Figure 14, the connection between the server and the client is routed through Dummynet that controls the bandwidth. With Dummynet, we can construct an evaluation environment to reproduce various network environments.

In actual broadcasting, we constructed a network environment where the server delivers data to several clients. In our evaluation, to show the effect of a broadcasting schedule, we created an environment in which the server delivers data to one client. In broadcasting, the server can concurrently deliver the same data to many clients in a constant network load. Therefore, the network load is not increased by adding clients.

When Dummynet controls the bandwidth, packet loss occurs. In this case, the setting bandwidth in Dummynet and the actual bandwidth are not necessarily the same. We calculated the bandwidth in Dummynet by measuring the amount of its received data per unit time.

7.2 Evaluation Items

We confirmed the availability of our proposed system to solve the following three problems: the effect on broadcasting schedules by the start delimiter header, the synchronization of timing for delivering data, and the sequential playback. Our proposed system compares its waiting and interruption times with a conventional system using the FB and BE-AHB methods.

7.3 Effect of Broadcast Scheduling by Start Delimiter

We evaluates several types of scheduling methods and confirmed the effectiveness of our proposed system. In it, processing time occurs in the separating and playing processes. This processing time is called overhead. First, we evaluated the waiting times under several amounts of segments. Next, we measured the waiting and interruption times under several consumption rates to evaluate the effect of overhead.

To compare simulation with actual environments, we calculated the waiting times in each environment. ``Actual'' denotes the waiting time under the actual environment, and ``simulation'' denotes it under the simulation environment.

In this evaluation, we used three types of scheduling methods: BE-AHB, FB, and an equally dividing method called ED. In the ED method, the settings for the data size of each segment and the available bandwidth of each channel are simple. When clients starts receiving segments concurrently using all the channels, the waiting times are not changed regardless whether the data are separated into several segments. We used the ED method to evaluate the effect of waiting times by dividing the data.

7.3.1 Effect of Number of Segments

Waiting time in this evaluation is the average waiting time from receiving a request from the browser to starting to play the data. We calculated waiting times under different numbers of channels. The result is shown in Figure 15. The horizontal axis is the number of channels, and the vertical axis is the waiting times. We used the ED, BE-AHB, and FB methods. The bandwidth of each channel was 1.5 Mbps, the consumption rate was 0.9 Mbps, and the playing time was 60 sec.

In Figure 15, since the simulation values are close to the actual values regardless of increasing the number of segments, the effect of waiting times by increasing processing times is low. In addition, in all the scheduling methods, the waiting times in the simulation environments are up 2% from those in the actual environment on average. Waiting times increase because of the overhead of other processes except for the separating process.

7.3.2 Consumption Rate and Interruption Time The evaluation result in Section 7.3.1 shows that interruptions in playing data might be caused by the overhead. Therefore, we evaluated how overhead affected interruptions.

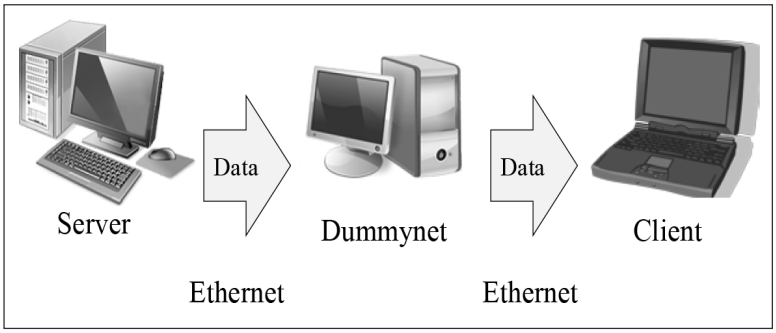

Figure 14. Assumed network environment

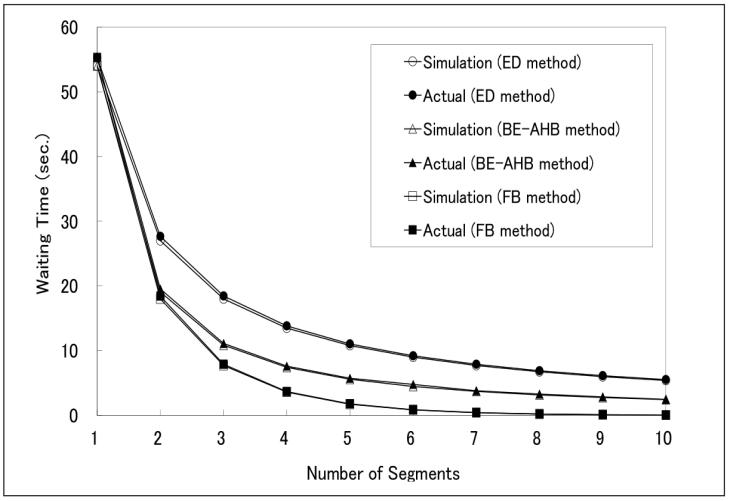

Figure 15. Number of channels and waiting time

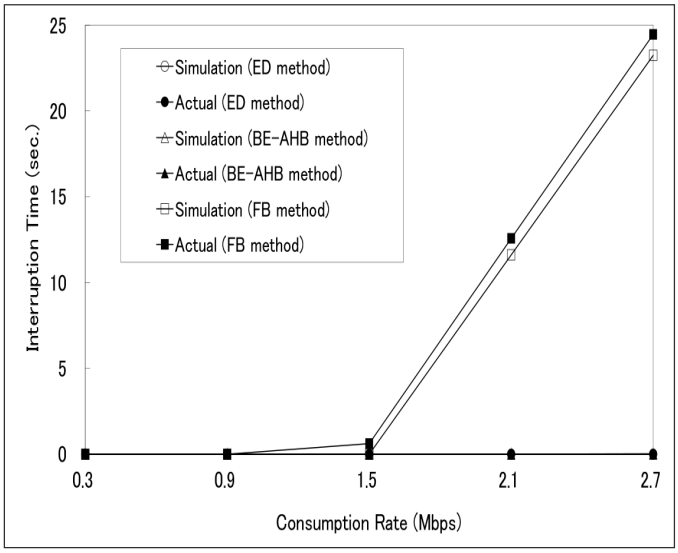

Figure 16. Consumption rate and interruption time

We calculates the interruption times under different consumption rates. The result is shown in Figure 16. The horizontal axis is the consumption rate, and the vertical axis is the interruption time. We use the ED, BE-AHB, and FB methods. The bandwidth of each channel was 1.5 Mbps, the number of segments was 5, and the playing time was 60 sec.

In Figure 16, when the consumption rate was 0.3 Mbps and 0.9 Mbps, no interruption time occurred in any scheduling methods. This is because clients completely received the next segment before they finish playing the current segment, and overhead did not influence the interruption times.

When the consumption rate was 1.5 Mbps, no interruption time occurred under the FB method in the actual environment. In the simulation environment, however, interruptions under the FB method did occur. Since the schedule of the FB method in the actual environment completely received the next segment before finishing to play the clients' received segment, no interruption occurred. On the other hand, in the simulation environment, interruptions in playing data occurred because the starting time of playing the next segment was delayed by the overhead.

When the consumption rate was 2.1 and 2.7 Mbps, interruption occurred under the FB method in both environments. The FB method does not address a case where the consumption rate exceeds the available bandwidth of each channel. Since clients do not completely receive the next segment until they finish playing the current segment, interruptions in playing data occurred.

7.4 Synchronization of Timing For Delivering Data

In a mechanism that synchronizes the timing of delivering data by considering additional information, we evaluated the performance of reducing interruption times with two scheduling systems: proposed and conventional systems.

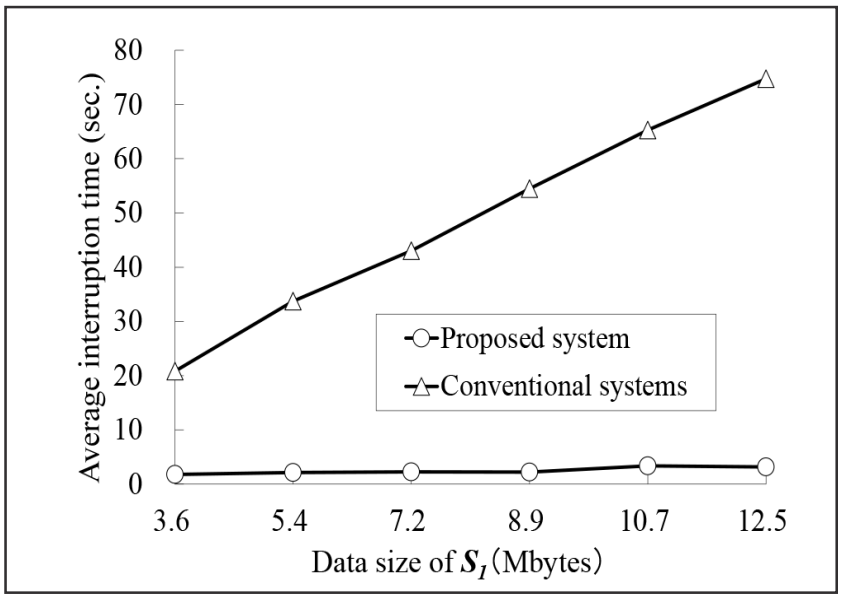

Figure 17. Data size of first segment vs. average interruption time

7.4.1 Data size of Segment and Interruption Time

The waiting time in our evaluation is the average waiting time from receiving a request from the browser to starting to play the data. We calculated the interruption times under a data size of S1. The result is shown in Figure 17. The horizontal axis is the data size of S1, and the vertical axis is the average interruption time for measuring three clients. We set several types of data size in S1 based on the playing time of the data. For example, when the data size of S1 was 3.6 Mbytes, its playing time was 180 sec. Except for the data size, the parameters are below:

- Scheduling method: FB method
- Number of segments and channels: two
- Channel bandwidth: 1.5 Mbps
- Consumption rate: 1.5 Mbps
- Data size of subsegments: 1460 bytes

In Figure 17, the average interruption time was 2.50 sec. under the proposed system. The average interruption time is constant as the data size of S1 increases. On the other hand, the average interruption time under the conventional systems increases as the data size of S1 increases. When the data size of S1 increases, the average interruption time under the proposed system is reduced more than in the conventional systems.

7.4.2 Number of Delivering Segments and Interruption Times

In division-based broadcasting systems, when the number of segments that muse be delivered repeatedly increases, the effect of additional information becomes large and interruption time change. We calculated the interruption times under the number of delivered S1. The result is shown in Figure 18. The horizontal axis is the number of

delivered $S1$, and the vertical axis is the average interruption time. The playing time of the data was 60 sec., and the number of segments and channels was 3.

In Figure 18, the average interruption time under the proposed system was 0.48 sec. The average interruption time remains constant as the data size of S increases. When the server starts delivering S, the average interruption time under the conventional systems greatly increases, because the difference of starting times between S and S is the largest. In addition, when the number of delivered S increases, this difference becomes smaller and the average interruption time is reduced. When the number of delivered S is between 2,100 and 2,200, the number of delivered S is reduced more than the number of delivered S and their difference again becomes large.

7.5 Sequential Playback

To evaluate the effect of reducing waiting times in our proposed system, we measured waiting times. We compared our proposed system and the conventional systems.

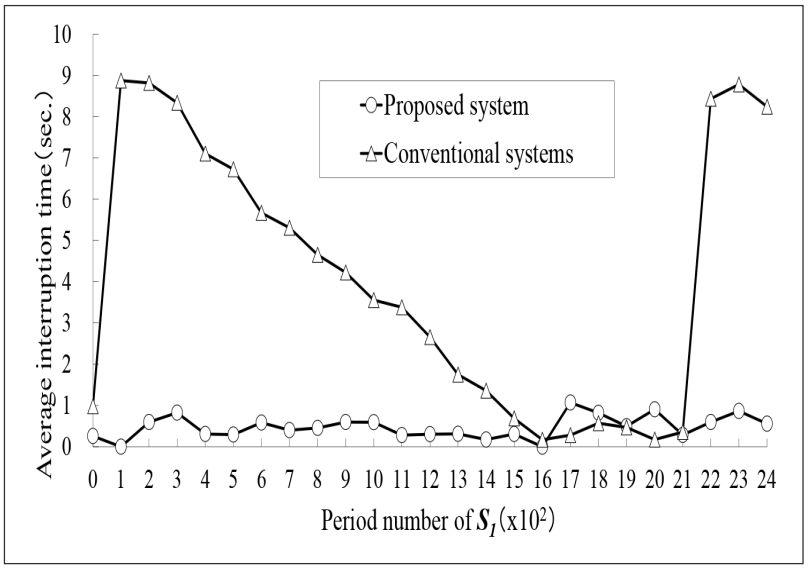

Figure 18. Period number in first segment vs. average interruption time

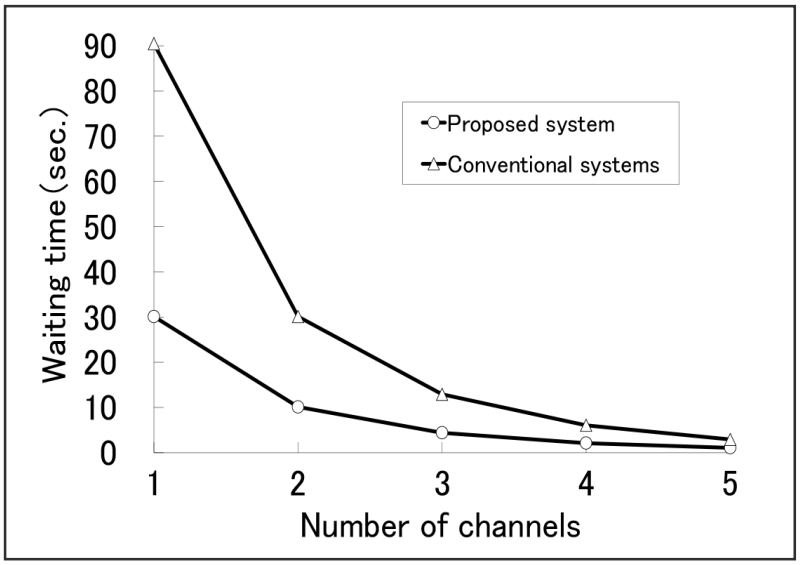

Figure 19. Number of channels vs. waiting time

The waiting time in our evaluation is the average waiting time from receiving a request from the browser to starting to play the data. We calculated the waiting times under different numbers of channels. The result is

shown in Figure 19. The horizontal axis is the number of channels, and the vertical axis is the waiting time. We used the FB method in this evaluation. The bandwidth of each channel was 1.5 Mbps, the consumption rate was 1.5 Mbps, and the playing time was 60 sec. The minimum subsegment data size changes based on the type of movie. The data size of the movie was 20,625 Bytes.

In Figure 19, the average waiting time under the proposed system was reduced more than that under the conventional systems in all cases. Since our proposed system uses sequential playback, it reduces the waiting times more than the conventional systems that use a download type.

For example, when the number of channels is two, the average waiting times under the proposed system are 10.1 sec. and 30.1 sec. under the conventional systems. The average waiting time under the proposed system was reduced by 66.4% compared to the conventional systems.

8. Conclusion

In this paper, we proposed a scheme to solve such problems as additional information that increases waiting times before playing data as well as interruption times while playing them. In addition, we designed and implemented a division-based broadcasting system. In our evaluation, we confirmed that our proposed system reduced waiting and interruption times. For example, when the number of channels was two, the average waiting times under our proposed system ware 10.1 sec. and 30.1 sec. under the conventional systems. The average waiting time under the proposed system was reduced by 66.4% compared to the conventional system.

In the future, we will realize a division-based broadcasting system that can automatically select the optimal scheduling method. Also, we will study a system that reduces communication traffic and the server's processing load by changing the delivery mode.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 26730059 and 15H02702. In addition, this work was partially supported by Strategic Information and Communications R&D Promotion Programme (SCOPE) and a grant from the Wesco Scientific Promotion Foundation.

References

[1] Gotoh, Y., Yoshihisa, T., Kanazawa, M. (2007). d-Cast: A Division Based Broadcasting System for IP Networks. In: Proceeding of IEEE International Conference on *Advanced Communication Technology* (ICACT'07), 1902-1907.

[2] Juhn, L.-S., Tseng, L. M. (1998). Fast data broadcasting and receiving scheme for popular video service. *IEEE Trans. Broadcasting*, 44 (1) 100-105.

[3] Yoshihisa, T., Tsukamoto, M., Nishio, S. (2005). A Broadcasting Scheme for Continuous Media Data with Restrictions in Data Division. *In*: Proceeding of IPSJ International Conference on Mobile Computing and Ubiquitous Networking (ICMU'05), 90-95.

[4] Juhn, L.-S., Tseng, L. M. (1997). Harmonic broadcasting for video-on-demand service. *IEEE Trans.Broadcasting*, 43 (3) 268-271.

[5] Hua, K. A., Bagouet, O., Oger, D. (2003). Periodic Broadcast Protocol for Heterogeneous Receivers. *In*: Proceedings of Multimedia Computing and Networking (MMCN '03), 5019 (1) 220-231.

[6] Yan, E. M., Kameda, T. (2003). An Efficient VOD Broadcasting Scheme with User Bandwidth Limit. *In*: Proceedings of ACM Multimedia Computing and Networking, 5019, 200-208.

[7] Lin, C., Ding, J. (2006). CAR: A low latency video-on-demand broadcasting scheme for heterogeneous receivers. *IEEE Trans. Broadcasting*, 52, 336-349.

[8] Zhu, X., Pan, R., Dukkipati, N., Subramanian, V., Bonomi, F. (2010). Layered internet video engineering (LIVE): Network-assisted bandwidth sharing and transient loss protection for scalable video streaming. *In*: Proceedings of IEEE INFOCOM, 226-230.

[9] Xiao, W., Agarwal, S., Starobinski, D., Trachtenberg A. (2010). Reliable Wireless Broadcasting with Near-Zero Feedback. *In*: Proceedings of IEEE INFOCOM, 2543-2551.

[10] Fountoulakis, N., Huber, A., Panagiotou, K. (2010). Reliable Broadcasting in Random Networks and the Effect of Density. *In*: Proceeding of IEEE INFOCOM, 2552-2560.

[11] Tantaoui, M., Hua, K., Do, T. (2004). BroadCatch: A periodic broadcast technique for heterogeneous video-on-demand. *IEEE Trans. Broadcasting*, 50 (3) 289-301.

[12] Shi, L., Sessini, P., Mahanti, A., Li, Z., Eager D.L. (2006). Scalable streaming for heterogeneous clients. *In*: Proceedings of ACM Multimedia, 22-27.

[13] Saparilla, D., Ross, K.W., Reisslein, M. (1999). Periodic broadcasting with VBR-encoded video. *In*: Proceedings of *IEEE INFOCOM*, 2 464-471.

[14] Janakiraman, R., Waldvogel, M., Xu, L. (2002). Fuzzycast: Efficient video-on-demand over multicast. *In*: Proceedings of IEEE INFOCOM 2002, 920-929.