Brossom: A P2P Streaming System for Webcast

Yusuke Gotoh[1], Kentaro Suzuki[2], Tomoki Yoshihisa[3], Hideo Taniguchi[1], Masanori Kanazawa[4]
[1]Graduate School of Natural Science and Technology
Okayama University, 3-1-1 Tsushima-naka
Kita-ku, Okayama, 700-8530, Japan
[2]BUFFALO INC., 30-20, 3 cho-me
O-su. Naka-ku, Nagoya, Aichi, 460-8315, Japan
[3]Cybermedia Center, Osaka University
5-1 Miho-gaoka, Ibaraki, Osaka, 567-0047, Japan
[4]The Kyoto College of Graduate Studies for Informatics
6 Tanaka-Monzen-cho, Sakyo-ku
Kyoto, 606-8225, Japan
{gotoh, tani}@cs.okayama-u.ac.jp, ke-suzuki@melcoinc.co.jp, yoshihisa@cmc.osaka-u.ac.jp, m_kanazawa@kcg.ac.jp

ABSTRACT: *Due to the recent spread of streaming delivery of movies, streaming using Peer-to-Peer (P2P) streaming technology has attracted much attention. In P2P streaming systems, to distribute the network load, peers from which the user receives data is selected at random. For this, clients have to wait until their desired data is delivered. Therefore, there are many researches to reduce the waiting time. However, because of the complexity of the implementation, they usually evaluate these methods using computer simulations. In actual environment, indeed, interruption time is not always reduced by increasing clients to deliver data. To evaluate the effectiveness of P2P streaming systems, it is important to implement an actual P2P streaming system. In this paper, we evaluate a P2P streaming system. By using our implemented P2P streaming system, we investigate situations that its system is effective. As a result of our evaluation, we confirmed that interruption time is reduced effectively.*

Keywords: P2P Streaming, P2P Network, P2P Streaming Evaluation, Digital Broadcast Systems, Internet Protocol

Received: 18 June 2011, Revised 29 July 2011, Accepted 4 September 2011

1. Introduction

Due to the recent popularization of digital broadcasting systems, delivering services using Internet Protocol (IP) networks has attracted much attention. In these services, clients connect and receive the data. In conventional server-client types, when the client requires the data, the server delivers the required data to it. Clients can play their desired data without interruption. On the other hand, with the popularization of the smart phone, bandwidth utilization increases rapidly. In conventional delivery, the server's load becomes higher as the number of clients increases. When necessary bandwidth surpasses available bandwidth, the server cannot deliver the data to new clients, so waiting time from requiring them to starting to play them occurs. Therefore, we focus on the research on delivery systems that can reduce necessary bandwidth.

In our research, we consider the streaming delivery using P2P networks. In P2P networks, there are several clients called peers, who require data and receive them from other peers. When a peer finishes receiving a part of the data, it can deliver it to other peers.

We have proposed several scheduling methods to reduce waiting time for selecting peers in P2P streaming systems. These researches often assume a simulation environment in which load balance for receiving and playing data does not occur. However, due to the complexity of implementation, these researches usually evaluate these methods using the computersimulation. In actual environments, the waiting time is not always reduced by increasing the number of peers. To evaluate the availability of P2P streaming systems, it is important to implement a P2P streaming system.

In this paper, by evaluating the P2P streaming system, we consider situations that its proposed system is effective. Our proposed system can introduce conventional scheduling methods, and we can construct a delivering system according to the type of clients. The contribution of the paper is that we confirm that our proposed method gives shorter interruption time than the conventional methods in our proposed system.

The remainder of the paper is organized as follows. Our assumed P2P streaming systems are explained in Section 2. Related works are introduced in Section 3. In Sections 4 and 5, we make the design and the implementation. The system is evaluated in Section 6, and discussed in Section 7. Finally, we conclude our paper in Section 8.

2. P2P Streaming

2.1 P2P Network Environment

We compare between server-client types with P2P streaming types. As shown in Figure 1, each peer connects to other peers in node relay-based webcast. In server-client types, since the server send data to clients, the load of server increases according to the number of peers. In P2P streaming types can distribute the load of delivering data using several peers.

Our assumed P2P network structure is shown in Figure 2. In P2P networks, there are two types of peers: request peer and provider peer. The request peer requires data and receives them from provider peers via the P2P network. The request peer finishes receiving the initial part of the data and plays it. When a request peer finishes receiving all the data, it becomes the provider peer. When a request peer wants streaming data from the provider peers, it receives data separated into data segments.

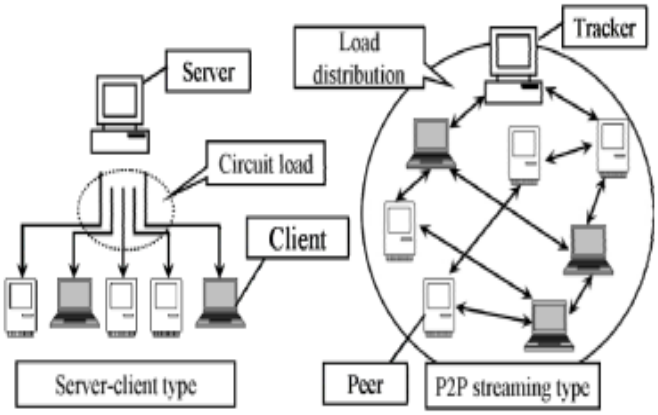

Figure 1. Comparison between server-client types with P2P streaming types

As shown in Figure 2, the tracker acquires information of peers periodically manage the status of them. In receiving data, the receive peer firstly acquires the list of IP address from the tracker. The receive peer selects several provider peers from the list and receives data. We explain the methods for selecting provider peers and receiving data segments in Section 4.

When the request peer receives a segment from provider peers, since waiting time occurs, users often feel annoyed. Therefore, in P2P streaming, we need to reduce waiting time occurred in receiving data.

2.2 Waiting time

In this subsection, we explain how the waiting time occurs. Since there is only one server, in the streaming of on-demand delivery, waiting time is in inverse proportion to the available bandwidth. However, in conventional P2P streaming, many provider peers exist. Since the request peer separates the data into several segments and delivers them, waiting time greatly changes based on the available bandwidth of each provider peer.

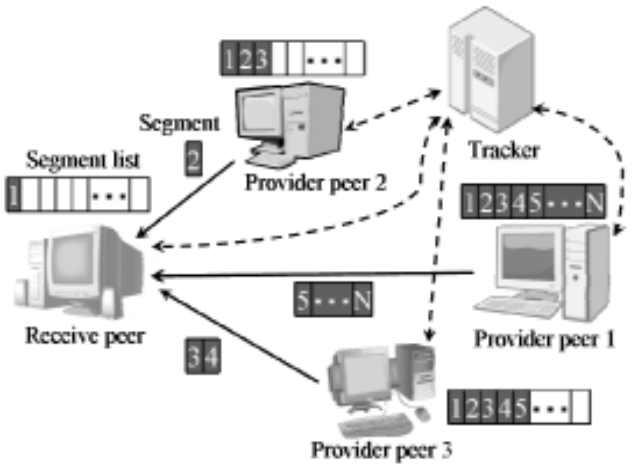

Figure 2. Our assumed P2P streaming system

In conventional methods, since the request peer chooses provider peers randomly to receive some of the data, waiting time increases based on the selected provider peers. For example, when the request peer wants data from the provider peers and receives segment S_{i+5j} (j = 0,…, 19) by channel b_i (i = 1, …, 5), the delivery schedule is shown in Figure 3.

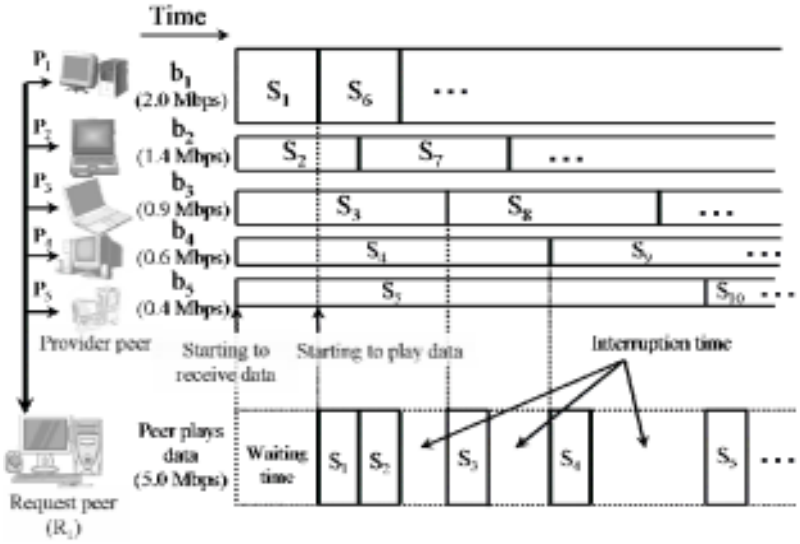

Figure 3. Example of a broadcast schedule under the simple method

The request peer is R_1, and the available bandwidth is 5.5 Mbps. Otherwise, provider peers are P_1, …, P_5. The bandwidth of b_1 is 2.0 Mbps, b_2 is 1.4 Mbps, b_3 is 0.9 Mbps, b_4 is 0.6 Mbps, and b_5 is 0.4 Mbps. The data consumption rate is 5.0 Mbps. When the data are separated into n segments, the separated segments are S_1, …, S_n. When the total data size is 256 MBytes and the data size of each segment is 25.6 MBytes, 256 / 25.6 = 10 . In Figure 3, when the provider peer receives S_1 for b_1, the waiting time is only the receiving time of S_1, which is $256\times 8/(2.0 \times 1000) = 1.0$.

In P2P streaming, users may experience annoyance when the interruption occurs between finishing time of the segment and starting time of next segment. In Figure 3, when the request peer receives 22 segments that are S_3, S_4, and S_{5j} (j = 1,…, 20), it plays them with interruption. In this case, total waiting time is 59.4 sec.

In P2P streaming, it is important that clients can play the data without interruptions until the end of the data. In conventional methods, waiting time is reduced based on available bandwidth of each provider peer.

3. Related Works

3.1 P2P Streaming Delivery

Several P2P delivering methods have been proposed [1]-[4]. In BitTorrent [5], [6], clients receive from peers the divided data of each segment called a piece. By providing a piece of the data to other peers, the client can receive other data from them. Provider peers whose available bandwidth is small can also deliver the data.

Gnutella [7], [8] is an application that shares data among clients. In this system, clients called servants send messages to other clients. However, since the system is not ready to streaming data, clients can not play the data until they have finished receiving them.

In P2P streaming delivery, several methods have been proposed [9]-[13]. Xu et al. proposed a P2P streaming concept where the request peer receives data from provider peers and analyzed the data capacity of P2P networks [14]. Shah et al. proposed a scheduling method to reduce waiting time [15] in which clients receive the divided data called a piece from peers using BitTorrent. In Narada method [16], when the provider peer delivers the data to request peers in P2P streaming, receiving time is reduced by reconstructing a P2P network and making a tree structure. Kageyama et al. proposed a replication method based on demand forecasting that prevents the loss of low-demand files in P2P networks [17]. CoolStreaming [18] is a data-driven relay route overlay network for P2P streaming. By using an efficient scheduling algorithm to fetch video segments from each peer, CoolStreaming achieves smooth video playback and good scalability.

PRIME [19] conducts that each P2P connection in a mesh streaming overlay should have roughly the same bandwidth to maximize the utilization of the available bandwidth in each provider peer. Zhang et al. proposed an optimal scheduling method for non-layered streaming, where the Min-Cost Flow Problem (MCFP) is employed for scheduling [20].

We previously proposed a scheduling method to reduce the waiting time in P2P streaming called the "Waiting time Reduction for P2P Streaming (WRPS)" method [21]. In the WRPS method, waiting time is effectively reduced by sequentially receiving the first segment of the data from a peer with large bandwidth. However, in the WRPS method, we do not assume the case where a provider peer delivers data to many request peers concurrently. In our paper, by designing and implementing P2P streaming systems, we consider situations in which our proposed system is effective. Since it can introduce conventional scheduling methods, we can construct a delivery system based on the type of clients.

3.2 Network Environment

StarBED [22] and Planetlab [23] are the testbed for emulating the network environment. These testbeds can evaluate many types of application protocols and devices. Our assumed P2P streaming system needs to design the communication protocol for delivering data. Therefore, in actual environment, we construct the network environment that the available bandwidth of each provider peer is set.

4. Design

4.1 Assumed Environment

In this research, we design a P2P streaming system called the "Broadcasting System with P2P Environments (Brossom)''. The detail design for Brossom is given below.

Our assumed system environment is summarized below. We use the terms explained in Section 2.

- The request peer receives data from one or more provider peers.
- Provider peers have all the segments of streaming data.
- The bandwidth is stable while delivering data.

4.2 System Configuration

We explain the detail of processes in Brossom.

4.2.1 Brossom Tracker

As shown in Figure 2, Brossom tracker plays a role of the server. Brossom tracker manages lists of contents and peers in P2P

networks. The list of contents is composed of a file name, a title, an abstract, a search tag, a data size, and the list of peers which has appropriate data. The list of peers is composed of the IP address, a port number, and available bandwidth of each peer.

4.2.2 R-Peer

The request peer called R-Peer requests the data to provider peers and receives the requested data from them. The R-Peer is connected to the Brossom tracker and receives the list of provider peers. Based on the list of provider peers, the R-Peer selects several provider peers and requests the data to them.

4.2.3 P-Peer

The provider peer called P-Peer delivers the requested data to the R-Peer. When the P-Peer replies the requests, it connects to the R-Peer. Based on the scheduling method to be used, the P-Peer delivers the requested data to the R-Peer.

4.3 Delivering Process of data

In this subsection, we explain a delivering process of data in Brossom. The detail of each process is given below.

4.3.1 Searching contents

The procedure of searching a content is shown in Figure 4. The user enters a retrieval word in a browser and requests the search to the Brossom tracker. The Brossom tracker refers the data of each peer in P2P networks and extracts the list of contents which is compatible with the retrieval word. The R-Peer receives the list from the Brossom tracker. The R-Peer can check the information of P-Peers which have its requested data.

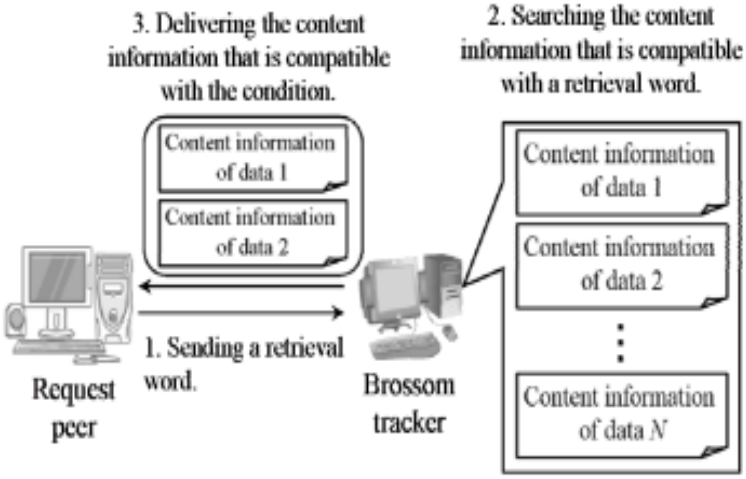

Figure 4. Structure of searching contents for Brossom

4.3.2 Acquirement of Peer Information

The procedure of acquiring peer information is shown in Figure 5. The R-Peer requests the information of P-Peers which have its requested data to the Brossom tracker. The Brossom tracker extracts the information of P-Peers which have its requested data and delivers it to the R-Peer. The R-Peer schedules contents based on information of P-Peers and requests the data to its selected P-Peers.

4.4 Processing in Actual Environment

Brossom uses a P2P streaming in actual environment. In actual environment, it is necessary to consider the interruption in playing data, load balance in delivering segments, and the overhead in receiving data.

4.4.1 Interruption in playing data

In the streaming type, since the R-Peer can play data after receiving a given amount of buffer size, we can reduce waiting time compared to the download type. When the buffer size is large, in the case where the receiving speed of data is later than the playing speed, the R-Peer can play the data during the playing time of it stored in its buffer. However, since the data size becomes large, waiting time from starting to receive data to starting to play them becomes long. In Brossom, the buffer size is that playing time of data is 10 sec.

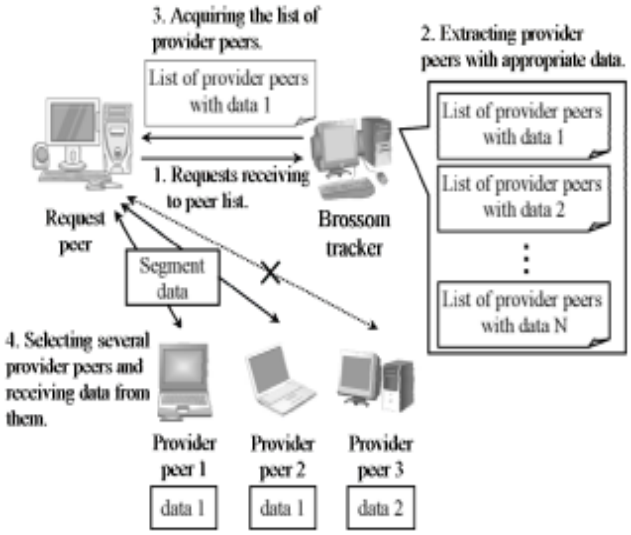

Figure 5. Structure of searching peers for Brossom

4.4.2 Load balance in delivering the segment data

In P2P streaming, the size of each segment data is set based on the software. The data size in Brossom is 128 KBytes. When the P-Peer delivers the large data at once, delivering time of the data using its available bandwidth becomes long. In Brossom, the data size becomes short by delivering the data by a segment. In the simulation environment, since load balance in delivering segments is not considered, waiting time is reduced by receiving data which is equally divided in each segment from P-Peers. On the other hand, in the actual environment, when the P-Peer delivers data which are equally divided in each segment from P-Peers, it loads them each time and divides each segment which the data size is 128 KBytes. In this case, since load balance of the P-Peer increases, processing time in delivering data increases.

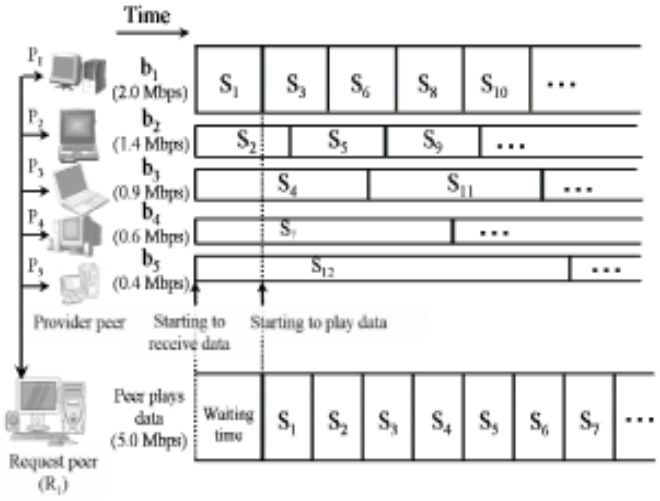

Figure 6. Example of a broadcast schedule under the WRPS method

4.4.3 Overhead in receiving data

In Brossom, each peer is connected by TCP/IP. In delivering data, since the P-Peer appends the header to the data packet, the data size increases, and delivering time becomes long.

4.5 Scheduling Methods

In our evaluation, we use the WRPS method. The WRPS method can reduce the waiting time by selecting peers and sequentially receiving the first bit of data from a peer that has large bandwidth. The detail of the scheduling process under the WRPS method is described in [21].

For example, in the WRPS method, suppose the case where the R-peer delivery the data which are separated from 100 segments. The delivery schedule is shown in Figure 6. The total data size is 256 MB and the data size of each segment is 25.6 MB. In Figure 6, when the P-peer receives *S1* for *b1*, waiting time is only the receiving time of *S1*, which is $256 \times 8 / (2.0 \times 1000) = 1.0$.

4.6 Definition of Simulation Environment

As explained in Section 1, conventional methods have evaluated in simulation environment. Our assumed simulation environment is summarized below:

- The process time in receiving data and playing them is zero.
- The performance of P2P streaming methods is evaluated by computer simulation.
- The waiting time is calculated by computation expression.

Request Peer (Streaming in one's Web Browser)

Provide Peer

Figure 7. Screen shot for Brossom

In simulation environment, we do not the process time in receiving data and playing them. In most P2P streaming methods, dividing the data into more segments causes further waiting time reduction.

5. Implementation

We implemented a P2P streaming system based on the system design explained in Section 4. A screen shot is shown in Figure 7. In our implementation, we use the WRPS method. The WRPS method can easily construct the delivery schedule in P2P streaming systems.

We implemented the system using Visual C#. In Brossom, we adopt TCP/IP. The system configuration of Brossom is shown in Figure 8. To control the available bandwidth of each P-Peer, we used an artificial bandwidth control machine called the FreeBSD Dummynet [24]. By setting the Dummynet between the R-Peer and P-Peers, Brossom tracker can control the available bandwidth of each P-Peer based on the network configuration.

Depending on the P2P streaming system, a network configuration in the actual environment can have many patterns. However, evaluating the performance of our proposed system for all of these patterns is not realistic. Therefore, in this paper, we use the network configuration which Brossom tracker controls the available bandwidth of each P-Peer using a Dummynet. Although practical network configurations do not always match these patterns, these are sufficient to show the effectiveness of our proposed system.

A Dummynet machine has four LAN ports, which three of them are 100 Mbps, and the rest of them is 1.0 Gbps. In this system, we use six machines, which one of them is a R-Peer, and five of them are P-Peers. In the measurement of communication speed, we use a software called the NetPerf [25].

The machine environment is shown in Table 1. We use the data format called the Flush Video (FLV).

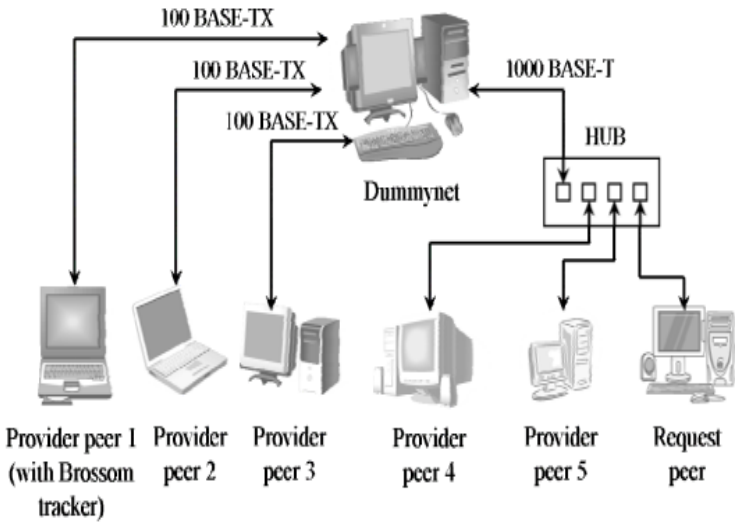

Figure 8. Experiment environment for Brossom

6. Evaluation

6.1 Basic Idea

In this section, we evaluate the performance of our proposed method. In our evaluation, we used the P2P streaming system. First, we explain the evaluation environment in Brossom. Next, we evaluate the comparison of the simulation environment with the actual environment.

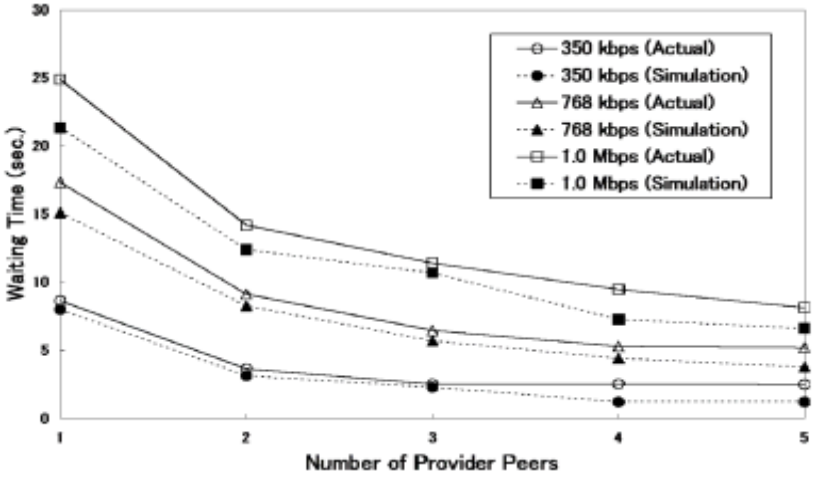

Figure 9. Number of receive peers and the average waiting time

6.2 Evaluation Environment

In our evaluation, the available bandwidths of each P-Peer are 480 kbps, 400 kbps, 320 kbps, 240 kbps, and 160 kbps. We use the three types of data consumption rate, which are 350 kbps, 768 kbps, and 1.0 Mbps. Playing time of data is 60 sec., and the data size of each segment is 128 Kbytes. The R-Peer can play data by finishing receiving the initial part of it, which is 10 sec. When the data in the buffer is empty, the R-Peer stops playing it and waits until finishing to receive the data, which is 10 sec. When the consumption rate is 350 kbps, the buffer size is 448 KBytes in playing data, which is 10 sec. When the consumption rate is 768 kbps, the buffer size of it is 983 KBytes. When the consumption rate is 1.0 Gbps, the buffer size of it is 1.0 GBytes. The data size of each segment is 128 KBytes.

R-Peer	CPU: Phenom 9600 2.3 GHz Memory: 4.0 GBytes OS: Windows XP Professional x64 Edition NIC: Realtek RTL8168/8110 Family Gigabit Ethernet NIC
P-Peer 1 (Available bandwidth) 480 kbps	CPU: CoreSolo U1500 1.3 GHz Memory: 1.0 GBytes OS: Windows Vista Business NIC: Intel (R) PRO/100 VE Mobile Connection
P-Peer 2 (Available bandwidth) 400 kbps	CPU: Pentium M 1.1 GHz Memory: 512 MBytes OS: Windows XP Professional NIC: Marvell Yukon 88E8055 PCI-E Gigabit Ethernet Controller
P-Peer 3 (Available bandwidth) 320 kbps	CPU: Pentium M 1.2 GHz Memory: 512 MBytes OS: Windows XP Professional NIC: Intel (R) PRO/1000 MT Mobile Connection
P-Peer 4 (Available bandwidth) 240 kbps	CPU: Pentium M 1.2 GHz Memory: 512 MBytes OS: Windows XP Professional NIC: Intel (R) PRO/1000 MT Mobile Connection
P-Peer 5 (Available bandwidth) 160 kbps	CPU: Pentium M 1.2 GHz Memory: 512 MBytes OS: Windows XP Professional NIC: Intel (R) PRO/1000 MT Mobile Connection
Dummynet	CPU: AMD (R) AthlonXP 1600+ 1.4 GHz Memory: 512 MBytes OS: FreeBSD 7.0-RELEASE NIC1: Intel (R) PRO/1000 MT Desktop Adapter Ethernet Controller NIC2: VIA VT6102 PCI 10/100Mb Ethernet Controller NIC3: 3Com EtherLink XL 10/100 PCI TX NIC NIC4: D-Link DFE-540TX ProFAST 10/100 Adapter

Table 1. PC specifications

6.3 Waiting Time

When the number of P-Peers increases, since the delivery schedule changes, playing time of the initial part of data becomes short. Therefore, it is important to evaluate the change of waiting time based on the number of P-Peers.

We calculate the average waiting time under some number of P-Peers. The result is shown in Figure 9. The horizontal axis is the number of P-Peers to which the R-Peer can connect. The vertical axis is the waiting time. To evaluate the comparison of the simulation environment with the actual environment, we calculate the waiting time in both environments. “Actual” denotes the waiting time under the actual environment, and “Simulation” denotes the waiting time under the simulation environment.

In this graph, as the number of P-Peers increases, waiting time is reduced because the available bandwidth of them increases. Also, in the simulation environment, delivering time of header information is not considered. Since the network delay in delivering data is not occurred, waiting time becomes short. For example, when the number of P-Peers is 3 and the consumption

rate is 1.0 Mbps, waiting time under the simulation environment and the actual environment is 10.7 and 11.4 sec., respectively. The average waiting time under the actual environment is 11.4 / 10.7 = 1.07 times compared to the simulation environment.

6.4 Interruption time and Number of Interruption

In playing the data, when the amount of it in the buffer becomes empty, the R-Peer waits until restarting to play it. If the interruption time and the number of interruption increases, users often feel annoyed. Therefore, in P2P streaming, evaluating the interruption time and the number of interruption is important.

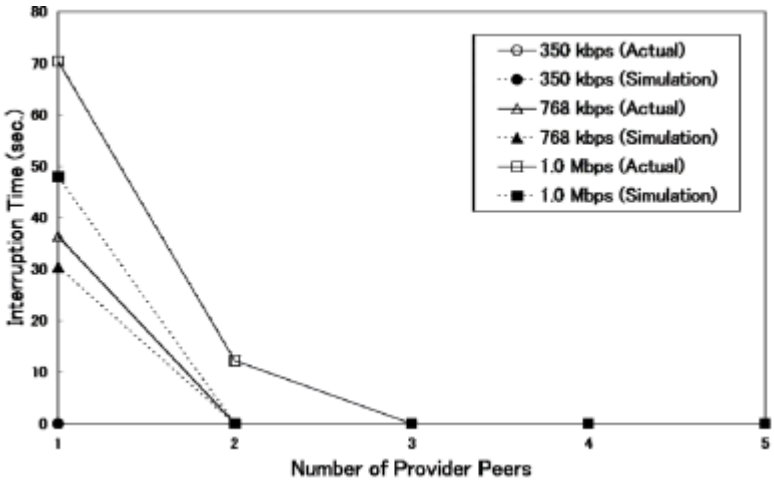

Figure 10. Number of receive peers and interruption time

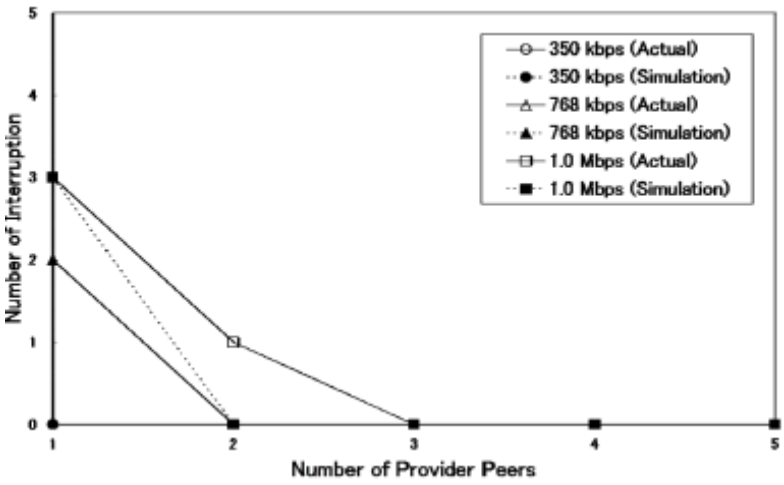

Figure 11. Number of receive peers and the number of interruption

The result is shown in Figures 10 and 11. The horizontal axis is the number of P-Peers to which the R-Peer can connect. The vertical axis is the interruption time in Figure 10, and the number of interruption in Figure 11. In this graph, as the number of P-Peers increases, the interruption time and the number of interruption is reduced. When the number of P-Peers increases, since the available bandwidth of them increases, the number of which the data in the buffer becomes empty decreases. Also, in Figures 10 and 11, when the consumption rate is 350 kbps, the interruption time is not occurred because the P-Peer which available bandwidth is larger than consumption rate delivers data.

6.5 Receiving time of data

We calculate the receiving time of data under the number of P-Peers. The result is shown in Figure 12. The horizontal axis is the number of P-Peers to which the R-Peer can connect. The vertical axis is the receiving time of segment data.

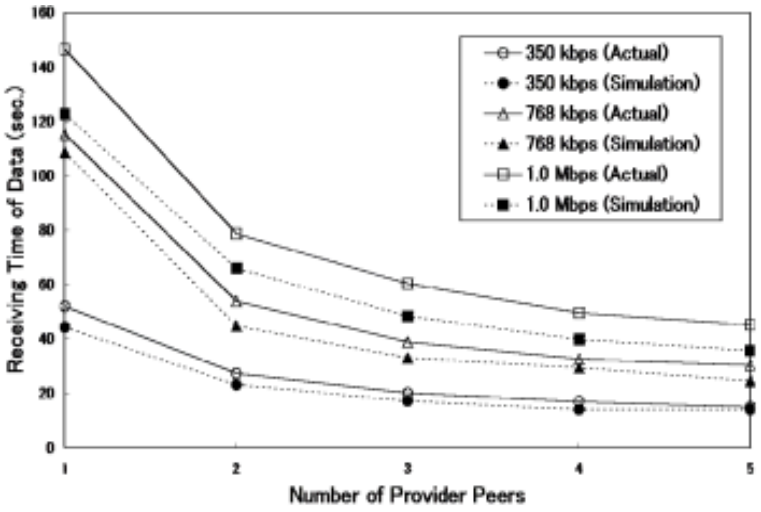

Figure 12. Number of receive peers and the receiving time of all segments

In this graph, as the number of P-Peers increases, receiving time of data is reduced. When the number of P-Peers increases, since the available bandwidth of them increases, receiving time of each P-Peer is reduced. For example, in the actual environment, when the consumption rate is 768 kbps, receiving time is 115.3 sec. under the case where the number of P-Peer is 1 and 30.3 sec. under the case where the number of P-Peer is 5. Therefore, receiving time under the case where the number of P-Peer is 5 is reduced (115.3 - 30.3) / 115.3 × 100 = 73.7% compared to the case where the number of P-Peer is 1.

7. Discussion

7.1 Effect of Interruption

When the waiting time in playing data occurs, playing time of them increases. If the R-Peer finishes receiving data until starting time of playing it, it can play the data without interruptions until the end of them. A watching player used in Brossom can play data by buffering them about 10 sec. of the playing time. Therefore, when the R-Peer finishes receiving all of the data within 60 + 10 = 70 sec., it can play them without interruptions.

7.2 Scheduling Method in Brossom

In the WRPS method, the delivery schedule is set before starting to receive the data. When the P-Peer which receiving rate is low delivers next segment during the interruption, since the interruption uncertainty occurs, receiving time increases, and the possibilities of occurring the interruption becomes higher.

7.3 Network environment in Brossom

In our simulation environment, since interruption time is calculated using computer simulation with several parameters, available bandwidth of each P-peer is constant. On the other hand, in our actual environment, since we set traditional TCP Reno, available bandwidth of each P-peer in starting to delivery data is small. After starting to delivery data, available bandwidth of P-peer increases gradually.

Several researches assume that the number of peers is more than several thousands. The aim of our paper is to confirm the effectiveness of the scheduling method which reduces interruption time. In our evaluation, using one R-peer and six P-peers, we confirmed that interruption time under the WRPS method is reduced than that under the simple method. Since several P2P streaming systems such as PeerStreaming [26] evaluate the effectiveness using few peers, our evaluation environment is effective.

When the number of R-peers increases, Brossom can spread demand across all P-peers by selecting them and receiving data. Also, by using the scheduling method, interruption time can be reduced compared with the simple method.

8. Conclusion

In this paper, we implemented the P2P streaming system called Brossom and proposed a new evaluation method. Our proposed system can introduce conventional scheduling methods, and we can construct a delivering system based on the type of clients. In P2P streaming, it is important to select P-Peers in receiving the data. We confirmed that interruption time is reduced effectively using our proposed method. Also, we considered about these points which is the interruption in playing data, the load balance in delivering them, and the overhead in receiving them.

In the future, we will make an implementation of client's environment such as prefetch and fast-forwarding. In addition, we need to evaluate the comparison the WRPS method with other scheduling methods.

9. Acknowledgement

This research was supported in part by The Japan Prize Foundation and JSPS Grant-in-Aid for Young Scientists (A) numbered 23680007. Also, this work was partially supported by a grant from the Okawa Foundation for Information and Telecommunications.

References

[1] Xiang, Z., Zhang, Q., Zhu, W., Zhang, Z., Zhang, Y.-Q. (2004). Peer-to-peer based multimedia distribution service, *IEEE Trans. on Multimedia*, 6 (2) 343-355.
[2] Hildrum, K., Kubiatowicz , J., Rao, S., Zhao, B. (2003). Distributed Object Location in a Dynamic Network, *In:* Proc. of 14th annual ACM symposium on Parallel algorithms and architectures. p.41-52.
[3] Liu, J., Rao, S.G., Li, B., Zhang, H. (2008). Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast, Proc. of IEEE, Special Issue on Recent Advances in Distributed Multimedia Communications, V.96, no. 1. p.11-24.
[4] Hei, X., Liu, Y., Ross, K.W. (2007). Inferring Network-Wide Quality in P2P Live Streaming Systems, *IEEE journal on Selected Areas in Communications*, 25 (9) 1640-1654.
[5] BitTorrent, http://www.bittorrent.com.
[6] Cohen, B. (2003). Incentives build robustness in BitTorrent, *In:* Proc. 1st Workshop on Economics of Peer-to-Peer Systems (P2PEcon'03).
[7] Gnutella, http://www.gnutella.com.
[8] Jnutella.org, http://www.jnutella.org.
[9] Guo, Y., Suh, K., Kurose, J., Towsley, D. (2003). A Peer-to-Peer on-demand streaming service and its performance evaluation, Proc. 2003 IEEE International Conference on Multimedia & Expo (ICME 2003), V.2, p.649-652.
[10] Tran, D., Hua, K., Do T. (2003). Zigzag: an efficient peer-to-peer scheme for media streaming, *In:* Proc. 22nd IEEE INFOCOM Conference, Vol.2, p.1283-1292.
[11] Guo, Y., Suh, K., Kurose, J., Towsley, D. (2003). P2Cast: Peer-to-peer Patching Scheme for VoD Service, *In:* Proc. 12th International Conference on World Wide Web (WWW), p.301-309.
[12] Padmanabhan V.N., Wang H., Chou P. (2003). Resilient peer-to-peer streaming, *In:* Proc. of 11th IEEE Int. Conf. on Network Protocols (ICNP'03), p.16-27.
[14] Cui Y., Nahrstedt K. (2003). Layered peer-to-peer streaming, Proc. of 13th int. workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03), p.162-171.
[15] Xu, D., Hefeeda, M., Hambrusch S., Bhargava B. (2002). On peer-to-peer media streaming, *In:* Proc. 22nd International Conference on Distributed Computing Systems (ICDCS2002), V.1, p.363-371.
[16] Shah, P., Paris, J.-F. (2007). Peer-to-Peer Multimedia Streaming Using BitTorrent, *In:* Proc. 26th International Performance of Computers and Communication Conference (IPCCC 2007), p.340-347.
[17] Chu, Y., Rao, S., Zhang, H. (2002). A Case for End System Multicast, *IEEE Journal on SAC, Special Issue on Networking Support for Multicast*, p.1456-1471.
[18] Kageyama, J., Kobayashi, M., Shibusawa, S., Yonekura, T. (2009). A file replication method based on demand forecasting in P2P networks, *In:* Proc. Second International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2009), p.268-274.
[19] Zhang, X., Liu, J., Li, B., Yum, T.P. (2005). CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming, *In:* Proc. of 24th IEEE Computer and Communications Societies (INFOCOM' 05), V.3, p.2102-2111.

[20] Magharei, N., Rejaie, R. (2007). PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming, *In*: Proc. of 26th IEEE Computer and Communications Societies (INFOCOM' 07), p.1415-1423.
[21] Zhang, M., Xiong ,Y., Zhang, Q., Yang, S. (2006). On the Optimal Scheduling for Media Streaming in Data-Driven Overlay Networks, *In:* Proc. IEEE Global Telecommunications Conference, GLOBESCOM 06.
[22] Gotoh, Y., Yoshihisa, T., Kanazawa, M. (2008). Method to Select Peers to Reduce Waiting Time in P2P Streaming Broadcasts, *In:* IADIS International Conference Telecommunications, Networks and Systems(TNS-CONF 2008), p.120-124.
[23] StarBED, http://www.starbed.org/.
[24] Planetlab, http://www.planet-lab.org/.
[25] Rizzo L., dummynet, http://info.iet.unipi.it/luigi/ip_dummynet/.
[26] NetPerf, http://www.netperf.org/netperf/.
[27] Li, J. (2004). PeerStreaming: A practical receiver-driven peer-to-peer media streaming system, Technical report, Microsoft.

Author biography

Yusuke Gotoh received the Bachelor's degree from Okayama University, Okayama, Japan, in 2005, and Master's and Doctor's degrees from Kyoto University, Kyoto, Japan, in 2007 and 2009, respectively. Since April 2009, he has been an Assistant Professor at Department of Information Technology, Faculty of Engineering, Okayama University, Okayama, Japan. From April 2011, he is a Visiting Researcher at La Trobe University, Melbourne, Australia, as a JSPS Postdoctoral Fellow for Research Abroad. His research interests include broadcast computing, CDN, and scheduling.

Kentaro Suzuki received the Bachelor's degree from Ritsumeikan University, Shiga, Japan, in 2007, and Master's degree from Kyoto University, Kyoto, Japan, in 2009. Since April 2009, he has been a researcher at BUFFALO INC. His research interests include network architecture and P2P technology.

Tomoki Yoshihisa received the Bachelor's, Master's, and Doctor's degrees from Osaka University, Osaka, Japan, in 2002, 2003, and 2005, respectively. From 2005 to 2007, he was a Research Associate at Kyoto University. In January 2008, he joined the Cybermedia Center, Osaka University as an Assistant Professor, 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 wearable computing.

Hideo Taniguchi received the BE degree in 1978, the ME degree in 1980, and the PhD degree in 1991, all from the Kyushu University, Fukuoka, Japan. In 1980, he joined NTT Electrical Communication Laboratories. In 1988, he moved from Research and Development Headquarters, NTT DATA Communications Systems Corporation. He had been an Associate Professor of Computer Science at Kyushu University since 1993. He has been a Professor of Faculty of Engineering at Okayama University since 2003. His research interests include operating system, real-time processing and distributed processing. He is the author of Operating Systems (Shoko Publishing Co. Ltd), etc. He is a member of the Information Processing Society of Japan (IPSJ), the Institute of Electronics, Information and Communication Engineers (IEICE), and ACM.

Masanori Kanazawa received ME from Kyoto University in 1971. From 1972, he was an Assistant Professor, Associate Professor and Professor of Data Processing Center, Kyoto University. From 1981, he was a Professor of Academic Center for Computing and Media Studies, Kyoto University. He is now a Professor of The Kyoto College of Graduate Studies for Informatics. He was also obtained a PhD degree in Engineering from Kyoto University. His current research interests include high-performance computing, grid computing and performance evaluation. He is a member of IPSJ and ACM.