Design and Implementation of P2P Streaming Systems for Webcast

Yusuke Gotoh[1], Kentaro Suzuki[2], Tomoki Yoshihisa[3], and Masanori Kanazawa[4]

[1] Graduate School of Natural Science and Technology, Okayama University, Okayama, Japan
gotoh@cs.okayama-u.ac.jp
[2] BUFFALO INC., Nagoya, Japan
ke-suzuki@melcoinc.co.jp
[3] Cybermedia Center, Osaka University, Osaka, Japan
yoshihisa@cmc.osaka-u.ac.jp
[4] The Kyoto College of Graduate Studies for Informatics, Kyoto, Japan
bwv147@jsbach.mbox.media.kyoto-u.ac.jp

Abstract. Due to the recent spread of different styles of watching movies, streaming using Peer-to-Peer (P2P) technology has attracted great attention. In P2P streaming systems, to distribute the network load, since peers from which the user receives data are selected at random, clients have to wait until their desired data are delivered. Therefore, many researches are attempting to reduce the waiting time. However, due to the complexity of implementation, they usually evaluate these methods using machine simulations. In actual environments, interruption time is not always reduced by increasing the number of clients who deliver data. To evaluate the availability of P2P streaming systems, implementing a P2P streaming system is crucial. In this paper, we design and implement a P2P streaming system. With our implemented system, we consider situations in which the proposed system is effective.

Keywords: Broadcasting, Waiting time, Streaming, Peer-to-peer.

1 Introduction

Recently, delivery services using Internet Protocol (IP) networks have attracted much attention and are changing how we watch movies. In these services, clients connect and receive data. In on-demand type, each client demands data from a server, who replies to the demand and delivers the demanded data. Although clients can get their desired data immediately, the server's load becomes higher as the number of clients increases. When necessary bandwidth surpasses available bandwidth, the server cannot deliver data to new clients, so the waiting time from starting to receive them to starting to play them increases.

In this paper, we consider streaming delivery using P2P networks. In P2P networks, there are several clients called peers, who demand data and receive them from other peers. When a peer finishes receiving all the data, it can deliver them to other peers.

We previously proposed several scheduling methods to reduce waiting time for selecting peers on 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, they usually evaluate these methods using machine simulation. In actual environments, waiting time is not always reduced by increasing the number of peers. To evaluate the availability of P2P streaming systems, implementing a P2P streaming system is important.

In this 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.

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

2 Related Works

Several P2P delivering methods have already been proposed [1,2,3]. In BitTorrent [4], clients receive from peers the divided data of each segment called a piece. By providing a piece of data to other peers, clients can receive other data from them. Provider peers whose available bandwidth is small can also deliver data. Since many provider peers deliver popular content, many peers can receive it on P2P networks.

Several P2P streaming methods have also been proposed [5,6,7]. Shah et al. proposed a scheduling method to reduce waiting time [8], in which clients receive the divided data of each segment called a piece from peers using BitTorrent. We previously proposed a scheduling method to reduce waiting time in P2P streaming called the “Waiting time Reduction for P2P Streaming (WRPS)” method [9]. In this method, waiting time is effectively reduced by sequentially receiving the first segment of data from a peer with large bandwidth. However, WRPS does not suppose the case where a provider peer concurrently delivers data to many request peers. Since the number of channels is reduced whose available bandwidth is the same as the consumption rate, waiting time increases.

3 P2P Streaming

In this section, we explain P2P streaming systems.

3.1 Peer Configuration

Our assumed P2P network structure is shown in Figure 1. In P2P networks, there are two types of peers: request and provider. In Figure 1, the request peer

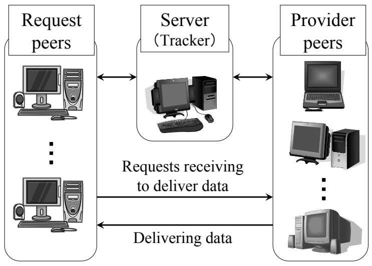


Fig. 1. Network structure in P2P streaming

is set in the center of the network and is connected to many provider peers. It demands data and receives them from provider peers using the P2P network. The request peer finishes receiving the initial part of the data and plays it. When the request peer finishes receiving all the data, it becomes the provider peer. When the request peer wants data from the provider peers, it receives data separated into segments.

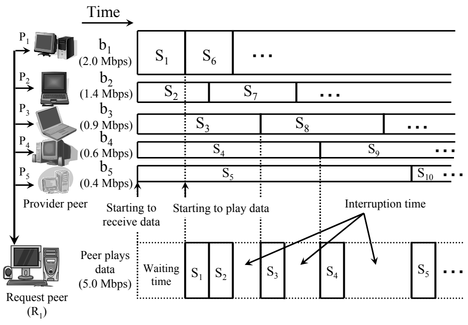
When the request peer receives a segment from provider peers, waiting time occurs, so users often feel annoyed. Therefore, in P2P streaming, we need to reduce the waiting time that occurs when receiving data.

3.2 Mechanism for Waiting Time Generation

In this subsection, we explain the mechanism for waiting time generation. In conventional methods, since the request peer chooses provider peers randomly to receive some parts 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} $(j = 0, \cdots, 19)$ by channel b_i $(i = 1, \cdots, 5)$, the delivery schedule is shown in Figure 2. The request peer is R_1, and the available bandwidth is 5.5 Mbps. Provider peers are $P_1, \cdots, P_5$. 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 $S_1, \cdots, S_n$. When the total data size is 256 MB and the data size of each segment is 25.6 MB, $n = 256/25.6 = 10$. In Figure 2, when the provider peer receives S_1 for b_1, the waiting time is only the time to receive S_1, which is $256 \times 8/(2.0 \times 1000) = 1.0$ sec.

In P2P streaming, users may become annoyed when an interruption occurs between the finishing time of a segment and the starting time of the next segment. In Figure 2, when the request peer receives 22 segments, S_3, S_4, S_{5j} $(j = 1, \cdots, 20)$, it plays them with interruption. In this case, the total waiting time is 59.4 sec.

In P2P streaming, clients must be able to play the data without interruptions until the end. In conventional methods, waiting time is reduced based on the available bandwidth of each provider peer. For example, in the WRPS method [9], suppose the case where the request peer delivers data that are



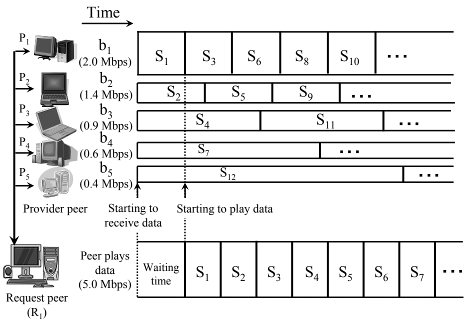


Fig. 2. Example of delivery schedule under simple method

Fig. 3. Example of delivery schedule under WRPS method

separated from 100 segments. The delivery schedule is shown in Figure 3. The WRPS method reduces waiting time by sequentially receiving the first bit of data from a peer with large bandwidth. 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 $256 \times 8/(2.0 \times 1000) = 1.0$.

3.3 Definition of Simulation Environment

Our assumed simulation environment is summarized below:

– The process time for receiving data and playing them is zero.
– In the simulation environment, we evaluate the waiting time and the interruption time by the computation simulation.
– Waiting time is calculated by computation expression.

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

4 Design

In this research, we design a P2P streaming system called the "Content Delivery System for P2P Streaming (DeSPerS)". Its design details are given below.

4.1 Assumed Environment

Our assumed system environment is summarized below:

– The request peer receives data from one or more provider peers.
– Provider peers have all data segments.
– Bandwidth is stable while delivering data.

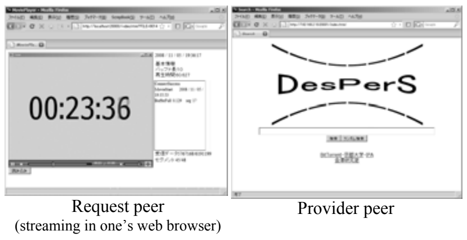


Fig. 4. Screenshot for DeSPerS

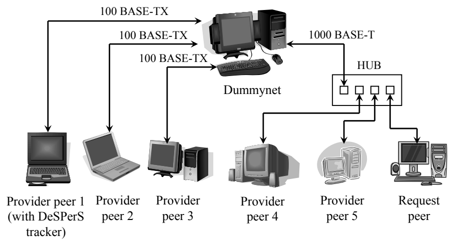


Fig. 5. Experiment environment for DeSPerS

4.2 System Configuration

In this subsection, we explain the details of the processes in DeSPerS. As shown in Figure 1, a DeSPerS tracker plays the role of the server and manages the lists of contents and peers in the P2P networks. The list of contents is composed of a file name, a title, an abstract, a search tag, data size, and a list of the peers with appropriate data. The list of peers is composed of an IP address, a port number, and the available bandwidth of each peer.

DeSPerS uses P2P streaming in actual environments, where the following must be considered: the interruption in playing data, the load balance in delivering segments, and the overhead in receiving the data.

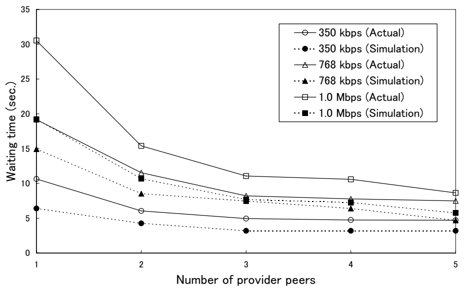
Interruption in playing data
In streaming delivery, since request peers can play data after receiving a given amount of buffer size, we can reduce the waiting time compared to the download type. In DeSPerS, the buffer size of the playing time of the data is 10 sec.

Load balance in delivering segment data
In P2P streaming, the size of each segment of data is set based on the software. The data size in DeSPerS is 128 KBytes. When a provider peer delivers a large amount of data at once, the delivery time using its available bandwidth becomes long. In DeSPerS, the data size is shortened by delivering the data in segments.

Overhead for receiving the data
In DeSPerS, each peer is connected by TCP / IP. For delivering data, since the provider peer appends a header to the data packet, the data size increases, and delivering time becomes longer.

5 Implementation

We implemented a P2P streaming system based on the system design. A screenshot is shown in Figure 4. In our implementation, we use a WRPS method that can easily construct a delivery schedule.



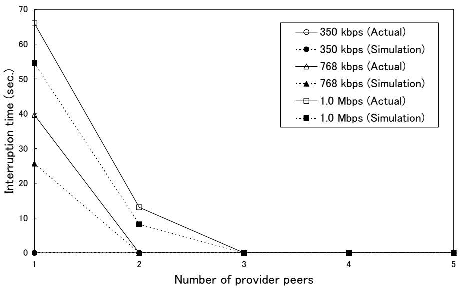


Fig. 6. Number of provider peers and average waiting time

Fig. 7. Number of provider peers and interruption time

The system configuration of DeSPerS is shown in Figure 5. To control the available bandwidth of each provider peer, we used an artificial bandwidth control machine called FreeBSD Dummynet [10]. By setting the Dummynet between the request and provider peers, the DeSPerS tracker can control the available bandwidth of each provider peer based on the network configuration.

Depending on the P2P streaming system, network configuration in actual environments can have many patterns. However, evaluating the performance of our proposed system for all of these patterns is not realistic. Therefore, we used a network configuration in which the DeSPerS tracker controls the available bandwidth of each provider peer using a Dummynet. In this system, we used six machines: one request peer, and five provider peers.

6 Evaluation

In this section, we evaluate the P2P streaming system.

6.1 Evaluation Environment

In our evaluation, the available bandwidth of each provider peer is 480, 400, 320, 240, and 160 kbps. We used the following data consumption rates: 350 kbps, 768 kbps, and 1.0 Mbps. The playing time of the data was 60 sec., and the data size of each segment was 128 Kbytes. The request peer can play the data after it finishes receiving the initial part, which is 10 sec. When there is no more data in the buffer, the request peer stops playing them and waits until it finishes receiving the data, which takes 10 sec. The data size of each segment is 128 KBytes.

6.2 Waiting Time

We calculated the average waiting time under different numbers of provider peers. The result is shown in Figure 6. The horizontal axis is the number of provider peers to which the request peer can connect. The vertical axis is the waiting time. To compare the simulation environment with the actual environment, we calculated the waiting time in each environment. “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 provider peers increases, waiting time is reduced because their available bandwidth increased. In the simulation environments, the delivery times of the header information are not considered. Since network delay for delivering data does not occur, waiting time is shortened. For example, when the number of provider peers is three, the waiting time under the simulation and actual environment is 7.7 and 11.1 sec., respectively. The average waiting time under the actual environment is $11.1/7.7 = 1.44$ times compared to the simulation environment.

6.3 Interruption Time

We calculated the interruption time under various numbers of provider peers. The result is shown in Figure 7. The horizontal axis is the number of provider peers to which the request peer can connect. The vertical axis is the interruption time in Figure 7. In this graph, as the number of provider peers increases, interruption time is reduced. When the number of provider peers increases, since their available bandwidth increases, the number of empty data in the buffer decreases. Also, in Figure 7, when the consumption rate is 350 kbps, interruption time does not occur because the provider peer whose available bandwidth is larger than the consumption rate delivers the data.

7 Discussion

7.1 Effect of Interruption

When an interruption in playing data occurs, playing time increases. If the request peer finishes receiving the data before the starting time of playing it, it can play all of the data without interruptions. In DeSPerS, we can play a piece of data by buffering it for about 10 sec. of playing time. Therefore, when the request peer finishes receiving all of the data within $60 + 10 = 70$ sec., it can play the data without interruptions until the end.

7.2 Scheduling Methods in DeSPerS

In the WRPS method, the delivery schedule is set before starting to receive data. When the provider peer whose receiving rate is low delivers the next segment during the interruption, since an interruption might occurs, receiving time increases, and the possibilities of an interruption occurring become higher.

8 Conclusion

In this paper, we designed and implemented a P2P streaming system. Our proposed system can introduce conventional scheduling methods, and we can construct a delivering system based on the type of clients. Also, we considered these points: interruption of playing data, load balance for delivering data, and overhead for receiving them.

In the future, we will make implementations for client environments such as prefetch and fast-forwarding. In addition, we need to compare WRPS with other scheduling methods.

Acknowledgment

This research was supported in part by Grant-in-Aid for Scientific Research (A) numbered 20240007 and a grant for Kyoto University Global COE Program, "Informatics Education and Research center for Knowledge-Circulating" from the Japanese Ministry of Education, Culture, Sport, Science and Technology of Japan.

References

1. Xiang, Z., Zhang, Q., Zhu, W., Zhang, Z., Zhang, Y.-Q.: Peer-to-peer based multimedia distribution service. IEEE Trans. on Multimedia 6(2), 343–355 (2004)
2. Liu, J., Rao, S.G., Li, B., Zhang, H.: Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast. Proc. of IEEE, Special Issue on Recent Advances in Distributed Multimedia Communications 96(1), 11–24 (2008)
3. Hei, X., Liu, Y., Ross, K.W.: Inferring Network-Wide Quality in P2P Live Streaming Systems. IEEE journal on Selected Areas in Communications 25(9), 1640–1654 (2007)
4. Cohen, B.: Incentives build robustness in BitTorrent. In: Proc. 1st Workshop on Economics of Peer-to-Peer Systems, P2PEcon 2003 (2003)
5. Tran, D., Hua, K., Do, T.: Zigzag: an efficient peer-to-peer scheme for media streaming. In: Proc. 22nd IEEE INFOCOM Conference, vol. 2, pp. 1283–1292 (2003)
6. Guo, Y., Suh, K., Kurose, J., Towsley, D.: A Peer-to-Peer on-demand streaming service and its performance evaluation. In: Proc. 2003 IEEE International Conference on Multimedia & Expo (ICME 2003), vol. 2, pp. 649–652 (2003)
7. Xu, D., Hefeeda, M., Hambrusch, S., Bhargava, B.: On peer-to-peer media streaming. In: Proc. 22nd International Conference on Distributed Computing Systems (ICDCS 2002), vol. 1, pp. 363–371 (2002)
8. Shah, P., Paris, J.-F.: Peer-to-Peer Multimedia Streaming Using BitTorrent. In: Proc. 26th International Performance of Computers and Communication Conference (IPCCC 2007), pp. 340–347 (2007)
9. Gotoh, Y., Yoshihisa, T., Kanazawa, M.: Method to Select Peers to Reduce Waiting Time in P2P Streaming Broadcasts. In: IADIS International Conference Telecommunications, Networks and Systems 2008 (TNS-CONF 2008), pp. 120–124 (2008)
10. Rizzo, L.: "dummynet", http://info.iet.unipi.it/~luigi/ip_dummynet/