Cache-based Adaptive Mechanisms for Media Streaming on Information Distributed Systems

Masahiro Sasabe

Department of Information Networking
Graduate School of Information Science and Technology
Osaka University

January 2006

Preface

With the growth of computing power and the proliferation of the Internet, media distribution services have widely diffused. However, with the current Internet, the major transport mechanism is still only the best effort service without no guarantees of communication qualities. Furthermore, the media streaming introduces much load into the network compared with other typical applications based on HyperText Transfer Protocol (HTTP) or File Transfer Protocol (FTP). From the viewpoint of users, high levels of Quality of Service (QoS), such as high-quality and continuous media play-out, are requested to the services. In addition, media qualities requested by users are heterogeneous and media popularity becomes diverse in the current Internet. Therefore, an effective media distribution system must be accomplished, which can provide users with media streams in a continuous way while suppressing load on the system.

To accomplish effective media streaming, routers and/or end systems require some control mechanisms. Integrated Services (IntServ) and Differentiated Services (DiffServ) are mechanisms to provide users with QoS guarantees of end-to-end communications. Multicast enables a server to distribute a media stream to many users simultaneously. However, the current Internet does not sufficiently support these mechanisms due to the deployment problem of routers that enable such functions. On the other hand, rate control of sending data, such as TCP Friendly Rate Control (TFRC), and caching of media data can be categorized as mechanisms at end

systems. We can easily employ these mechanisms because they do not rely on such network equipments. Furthermore, they can flexibly adapt to requirements of each user and its application.

In this thesis, we focus on the caching mechanisms to accomplish media streaming that can adapt to heterogeneity of user demands and network changes. At first, we discuss proxy caching mechanisms with a media quality adjustment in a client-server architecture. After introducing the overview of our system, we propose three mechanisms: media retrieval mechanism, media prefetching mechanism, and cache replacement mechanism. Through several simulation experiments, we evaluate our proposed mechanisms in terms of required cache buffer size, play-out delay, and media quality. Simulation results show that our system is effective in suppressing the play-out delay and reducing the required cache buffer size with a media stream of the desired quality. Specifically, waiting time for media play-out is less than 8 seconds even under a severe situation where the cache buffer size is limited to 20 Gbits. We further implement our proposed mechanisms on a real system. Through experiments, it is shown that our implemented system can continuously provide users with a high-quality and low-delay media service in accordance with the network condition.

Next, we examine the media streaming on unstructured Peer-to-Peer (P2P) networks where there is no server. There are several issues to resolve in accomplishing effective media streaming on unstructured P2P networks: scalability, continuous media play-out, adaptability to changes of system conditions. We first describe continuous and scalable media streaming on unstructured P2P networks. We propose scalable media search methods taking into account temporal order of reference to media data that means users usually play out a media stream from the beginning to the end. Then, we also propose media retrieval methods to accomplish continuous media play-out. Through several simulation experiments, we show that the proposed methods can provide users with continuous media play-out for popular media streams

while reducing search traffic to $\frac{1}{6}$ compared with a traditional method.

Then, we discuss adaptive and robust P2P media streaming. First, we propose a novel cache replacement algorithm that considers balance between supply and demand for media streams. Next, to improve the adaptability and robustness against changes of system conditions, such as network conditions, and peer departures, we further propose an adaptive media retrieval method and a robust media search method. Simulation results show that our proposed cache replacement algorithm can accomplish continuous media play-out independent of media popularity and adapt to changes in media popularity. Furthermore, we demonstrate that the adaptive media retrieval method can provide users with more continuous media play-out than our previous method. In addition, we find that the proposed search method is robust to peer departures.

Finally, we focus on the construction of a P2P overlay network to improve the user QoS in terms of search and retrieval time of files. After introducing several related works, we propose a construction method of a low-diameter and location-aware P2P overlay network where a peer can find many physically-close provider peers. We further propose a rewiring method to improve the structure of the constructed overlay network. Through evaluations in terms of diameter, physical distance between neighboring peers, and degree distribution of the overlay network, it is shown that our proposed method without rewiring can construct a low-diameter overlay network comparable with traditional method. In addition, we find that the rewiring method contributes to reduction of both the physical distance between neighboring peers and diameter of the overlay network. Especially, the proposed methods can accomplish the same logical reachability as the traditional method while reducing network load to 50%.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Professor Masayuki Murata of Graduate School of Information Science and Technology, Osaka University for his encouragement, valuable discussions, and meaningful advices throughout my studies.

I would like to express my deep gratitude to Professor Hirotaka Nakano and Professor Teruo Higashino of Graduate School of Information Science and Technology, Osaka University for serving as readers of my thesis committee.

I would like to sincere appreciation to President of Osaka University Hideo Miyahara for his helpful advices and continuous support.

I am deeply grateful to Associate Professor Naoki Wakamiya of Graduate School of Information Science and Technology, Osaka University for his continuous support, appropriate guidance, and invaluable direct advice.

In addition, I would like to thank Professor Koso Murakami and Professor Makoto Imase of Graduate School of Information Science and Technology, Osaka University for their support and teaching.

I am also indebted to Associate Professor Hiroyuki Ohsaki and Associate Professor Go Hasegawa of Graduate School of Information Science and Technology, Osaka University for their helpful comments and feedback.

I also deeply thank to Assistant Professor Shin'ichi Arakawa of Graduate School of Information Science and Technology, Osaka University for his invaluable advice.

I also wish to acknowledge my friends and colleagues for their continuous encouragement.

Finally, I heartily thank my parents and sister for their understanding and support.

List of Papers

Journal Papers

[1] M. Sasabe, Y. Taniguchi, N. Wakamiya, M. Murata, and H. Miyahara, "Proxy Caching Mechanisms with Quality Adjustment for Video Streaming Services," *IEICE Transactions on Communications Special Issue on Content Delivery Networks*, vol. E86-B, pp. 1849–1858, June 2003.

[2] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Effective Methods for Scalable and Continuous Media Streaming on Peer-to-Peer Networks," *European Transactions on Telecommunications*, vol. 15, pp. 549–558, Nov. 2004.

[3] M. Sasabe, N. Wakamiya, and M. Murata, "Adaptive and Robust P2P Media Streaming," *WSEAS TRANSACTIONS on COMMUNICATIONS*, vol. 4, pp. 425–430, July 2005.

Refereed Conference Papers

[1] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Proxy Caching Mechanisms with Video Quality Adjustment," in *Proceedings of SPIE International Symposium ITCom 2001*, vol. 4519, (Denver), pp. 276–284, Aug. 2001.

[2] M. Sasabe, Y. Taniguchi, N. Wakamiya, M. Murata, and H. Miyahara, "Implementation and Evaluation of Proxy Caching Mechanisms with Video Quality Adjustment," in *Proceedings of ITC-CSCC 2002*, vol. 1, (Phuket), pp. 121–124, July 2002.

[3] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Scalable and Continuous Media Streaming on Peer-to-Peer Networks," in *Proceedings of P2P 2003*, (Linköping), pp. 92–99, Sept. 2003.

[4] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Media Streaming on P2P Networks with Bio-inspired Cache Replacement Algorithm," in *Proceedings of Bio-ADIT 2004*, (Lausanne), pp. 380–395, Jan. 2004.

[5] M. Sasabe, N. Wakamiya, and M. Murata, "Adaptive Media Streaming on P2P Networks," in *Proceedings of ATNAC 2004*, (Sydney), pp. 549–556, Dec. 2004.

[6] M. Sasabe, N. Wakamiya, and M. Murata, "Adaptive and Robust P2P Media Streaming," *9th WSEAS International Conference on COMMUNICATIONS*, July 2005.

Non–Refereed Technical Papers

[1] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Caching Mechanisms for Proxy Cache with Quality Adaptation," *Technical Report of IEICE, (NS2001-53) (in Japanese)*, pp. 31–36, June 2001.

[2] Y. Taniguchi, M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Implementation and Evaluation of Proxy Caching Mechanisms with Video Quality Adjustment," *Technical Report of IEICE, (CQ2002-72) (in Japanese)*, pp. 41–46, July 2002.

[3] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Scalable Methods for Media Streaming on Peer-to-Peer Networks," *Technical Report of IEICE, (NS2003-101) (in Japanese)*, pp. 71–76, Sept. 2003.

[4] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Cache Replacement Algorithm for P2P Media Streaming," *Technical Report of IEICE, (SB-10-2)*, pp. SE–3–SE–4, Mar. 2004.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the growth of computing power and the proliferation of the Internet, media streaming services have widely diffused. However, with the current Internet, the major transport mechanism is still only the best effort service, which offers no guarantees of network bandwidth, delay, and packet loss probability. Consequently, the media streaming system cannot provide users with media streams in a continuous way. Furthermore, the media streaming introduces much load into the network compared with other typical applications based on HyperText Transfer Protocol (HTTP) [1] or File Transfer Protocol (FTP) [2]. This is because the media streaming needs to high network bandwidth to transfer the media data to users. For instance, the coding rate of a high-quality MPEG-2 media stream is 8 Mbps. Although MPEG-4 with a high compression ratio has been developed for the distribution in the Internet, it requires a few Mbps to provide users with a high quality media stream. Furthermore, the media streaming constantly induces heavy traffic into the network. Short clips, such as CM and the preview of a movie, are a few or minutes long while the length of a movie reaches a few hours. The number of users also affects the network load. Many users simultaneously join the media streaming system as well as the other web services.

To accomplish effective media streaming, routers and/or end systems require some control mechanisms. Integrated Services (IntServ) [3], Differentiated Services (DiffServ) [4], and multicast are control mechanisms supported by routers. IntServ and DiffServ are mechanisms to provide users with Quality of Service (QoS) guarantees. While IntServ completely allocates network resources to each user by using Resource reSerVation Protocol (RSVP) [5, 6], Diffserv classifies traffic in accordance with its priority. Multicast mainly represented by IP multicast is designed for use in live broadcasting in which many users simultaneously watch the same media stream. IP multicast constructs a multicast tree to deliver a media stream to users. The multicast tree consists of a media delivery server, IP multicast routers, and users. The root of the tree, that is the server, transmits the media stream to the IP multicast routers. Then, each IP multicast router forwards the received media stream to all of its children. As a result, the multicast is obviously more effective than unicasting the media stream for each user.

However, the current Internet does not sufficiently support these mechanisms due to the deployment problem of routers that enable such functions. On the other hand, rate control of sending data, such as TCP Friendly Rate Control (TFRC) [7], and caching of media data can be categorized as mechanisms at end systems. We can easily employ these mechanisms because they do not rely on such network equipments. Furthermore, they can flexibly adapt to requirements of each user and its application.

In this thesis, we mainly focus on the caching mechanism. Specifically, we discuss two kinds of approaches: proxy caching mechanisms on client-server architectures and caching mechanisms on Peer-to-Peer (P2P) architectures. At first, we propose proxy caching mechanisms for media streaming [8-12]. The proxy caching mechanism widely used in WWW systems offers low-delay delivery of data by means of "proxy server". A proxy server caches media data which have passed through it in its local buffer, called "cache buffer", then it provides the cached data to users on demand.

By applying proxy mechanisms to a media streaming system, high-quality and low-delay media distribution can be accomplished without introducing extra load on the system[13-20].

To accomplish continuous media streaming, there are several challenging tasks: heterogeneous user environments, changes of network conditions, and management of cached data. First, the heterogeneity of the Internet access, such as Fiber To The Home (FTTH) and Asymmetric Digital Subscriber Line (ADSL), is significant for the media streaming. Since narrow-band users cannot receive a high quality media stream in a real-time fashion, the proxy has to store a low quality media stream in its cache buffer to serve all users independent of their access connections. However, this approach apparently deteriorates the QoS of broad-band users. Furthermore, caching media streams of multiple qualities leads to the cache overflow. One way to solve these problem is the quality adjustment of the media stream on the proxy. There are two kinds of the quality adjustment: layered encoding and QoS filtering [21, 22].

The layered encoding transcodes the original media stream into multiple layers: a base layer and one ore more extension layers. By adjusting the number of the layers that are transmitted to users, the proxy can provide users with different quality media streams without transcoding. However, the degree of the quality adjustment is limited by the number of the layers. On the other hand, the QoS filtering techniques, such as frame dropping, low-pass, and re-quantization, are other schemes of the quality adjustment. Although these filtering techniques require more processing power compared with the layered encoding, they can flexibly adjust the media quality. The growth of the computing power enables the proxy server to conduct the QoS filtering in a real-time fashion. In this thesis, we employ the QoS filtering techniques. By applying the quality adjustment techniques, the proxy can provide users with different qualities of the media stream taking into account their access connections [13].

Next problem is changes of network conditions in terms of the available bandwidth

and Round Trip Time (RTT). To provide a media stream with users in a continuous way, the proxy needs to adapt the data sending rate in accordance with the network conditions. In most cases, media data are transmitted over UDP. Since UDP does not retransmit packets discarded at intermediate routers, it can quickly transfer data compared with TCP. In the case of media streaming, the packet retransmission is not necessarily meaningful because the media data should be received at a user before their corresponding play-out time. However, UDP cannot fairly coexist with the other traffic transmitted over TCP due to the lack of the rate control mechanism. To tackle this problem, we introduce TCP Friendly Rate Control (TFRC) protocol that enables UDP to behave in a TCP-friendly fashion. TFRC controls the data transmission rate based on the rate estimation method used in TCP. By adjusting the media quality in accordance with the sending rate estimated by TFRC, the proxy can provide users with the media stream of appropriate quality in a continuous way.

Finally, caching strategy that affects the cache hit probability is also significant to reduce the network load and improve the user QoS. We consider the identical characteristics of media streams: data size, quality, and temporal order of reference. The data size of a media stream is considerable huge compared with other objects, such as text and image. For the efficient use of the cache buffer and the network bandwidth, a media stream is divided into blocks. The proxy should retrieve a block of appropriate quality from a media server and store it into the cache buffer to prevent cache misses to the succeeding user requests. The last feature of the media streaming is that a user ordinarily plays out a media stream from the beginning to the end in a sequential order. By utilizing this temporal order of reference to media blocks, the proxy can predict what blocks will be requested by users in the near future. Based on this assumption, we also propose a prefetching mechanism and a cache replacement algorithm.

For further low-delay and high-quality media streaming, cooperation among proxies is also effective [23-25]. By exchanging information among the media server and the proxy cache servers, they can provider a user with a media stream from an appropriate server. However, their servers are statically located at various points in the network and have to process all requests coming in. It has been pointed out that such client-server model lacks scalability and stability against the number of users. To improve the scalability against the number of users and adaptability to various user demands, it is preferred that the number of hosts that provide media streams increases proportional to the number of users.

P2P is a new network paradigm in which hosts called peers directly communicate with each other and exchange information without the mediation of servers. Such distributed manner improves the system scalability against the increase of the users because traffic concentration to a specific point can be avoided. We expect that P2P architectures can support media streaming services based on the traditional client-server architectures. Typical examples of P2P systems are file-sharing, Application Layer Multicast (ALM), and Grid computing. By applying the P2P communication architectures, the media streaming can become more scalable to the number of users and more adaptive to heterogeneous user demands. There are two kinds of P2P media streaming systems: ALM and media streaming based on the file-sharing system.

ALM is a new multicast scheme in which peers function as the multicast routers targets the live media streaming. While the IP multicast requires to deploy multicast routers in the network to construct a multicast tree, ALM does not need extra resources by using peers as routers. However, peers dynamically join and leave during a session in ALM. In addition, peer-to-peer media delivery induces much load on the network because peers are generally located at the edges of the network. Many researchers have studied how to build an efficient multicast tree to solve these problems [26-33].

However, when user demands arise intermittently and peers request a variety of media streams, as in current P2P services, an efficient distribution tree cannot be composed. In this thesis, we discuss the on-demand media streaming based on the P2P file-sharing system[34, 35]. There are mainly three different architectures for the P2P file-sharing systems: centralized, decentralized-structured, and decentralized-unstructured. In a centralized model, such as Napster [36], meta servers maintain the information on shared files stored on peers. By sending a query to the servers, a peer can quickly find its desired file. However, this centralized architecture has a problem of a single point of failure. On the other hand, decentralized models enable users to exchange information and file directly with each other without the mediation of the server. In a decentralized-structured model, such as Chord [37], Pastry [38], Tapestry [39], and CAN [40], the location of shared files and the topology of a P2P network are tightly controlled based on the Distributed Hash Tables (DHT). Although DHT can improve the search efficiency, it requires to strictly manage the topology of a P2P network against the join and departures of peers. A decentralized-unstructured model, such as Gnutella [41] and KaZaA [42], has no correlation between the file location and the network topology. Because of this simple structure, the unstructured P2P systems are the most common in the current Internet. However, they also have several challenging tasks to improve the effectiveness of search because flooding, which is a typical search method, is not scalable to the number of peers.

We propose novel methods for on-demand media streaming based on the unstructured P2P file-sharing system [43-50]. From the viewpoint of service, we assume that a media streaming server can accomplish effective media streaming supported by our proposed P2P streaming system. In this thesis, we focus on to verify how high level of QoS can be accomplished by our P2P media streaming system. There are several issues to resolve in accomplishing effective media streaming on unstructured P2P networks: scalability, continuous media play-out, adaptability to changes of system

conditions. In our system, every peer participating in a service watches a media stream and deposits it in its local cache buffer. A media stream is divided into blocks for efficient use of network bandwidth and cache buffer. By retrieving blocks from other peers in time, a peer can watch a desired media stream. Since there is no server that manages information on peer and media locations, a peer has to find each block constituting a desired media stream by emitting a query message into the network. Other peers in the network reply to the query with a response message and relay the query to the neighboring peers. If a peer successfully finds a block cached in other peers, it retrieves it from one of them and deposits it in its local cache buffer. If there is no room to store the newly retrieved block, a peer has to perform replacement on cached blocks with it.

First, we tackle the problem of scalability in terms of search traffic. Flooding, in which a peer relays a query to every neighboring peer, is a powerful scheme for finding a desired media stream. However, it has been pointed out that the flooding lacks scalability because the number of queries that a peer receives significantly increases with the growth in the number of peers [51]. In particular, a block-by-block search by flooding apparently introduces much load on the network and causes congestion. To tackle this problem, we propose two scalable block search methods. Taking into account the temporal order of reference to media blocks, a peer sends a query message for a group of consecutive blocks. Then, the peer performs adaptive block search by regulating the search range based on the preceding search result.

Next, continuous media play-out is also important for users in media streaming services. To retrieve a block by its corresponding play-out time, we propose methods to determine an appropriate provider peer (i.e., a peer having a cached block) from search results by taking into account the network conditions, such as the available bandwidth and the transfer delay. By retrieving a block as fast as possible, the remaining time can be used to retrieve the succeeding blocks from distant peers.

Finally, we consider the adaptability and robustness to changes of system conditions, such as popularity of media streams, network condition, and peer departures. To improve the completeness of media play-out, we propose an effective cache replacement algorithm that takes into account the supply and demand for media streams. Since there is no server, a peer has to make conjectures about the behavior of other peers by itself. A peer estimates the supply and demand from P2P messages that it relays and receives from a flooding-based media search. Then a peer determines a media stream to discard to make room for a newly retrieved block. Furthermore, a peer also adapts to changes in the supply and demand of media streams. For this purpose, we propose a novel caching algorithm based on the response threshold model of division of labor and task allocation in social insects [52].

In biology, social insects, such as ants, also construct a distributed system [53]. In spite of the simplicity of their individuals, the insect society presents a highly structured organization. It has been pointed out that social insects provide us with a powerful metaphor for creating decentralized systems of simple interacting [53, 54]. In particular, a recently proposed model of division of labor in a colony of primitively eusocial wasps, based on a simple reinforcement of response thresholds, can be transformed into a decentralized adaptive algorithm of task allocation [52]. By regarding the replacement of media streams as a task, we propose a fully distributed and autonomous cache replacement algorithm which can adapt to changes in environments, i.e., the supply-to-demand, without parameter tunings.

Furthermore, in an actual situation, media streaming fails since peers participating a service occasionally leave a P2P network due to user's interactions or system failures. Network conditions including the available bandwidth and RTT also change dynamically. We newly propose a block retrieval method which dynamically switches provider peers based on the estimation of the available bandwidth and RTT. In addition, we extend our block search method to prepare for peer departures.

The structure of a P2P overlay network is also important because it determines the effectiveness of search in terms of network load and user QoS. An inefficient overlay network induces unnecessary and huge traffic into underlying physical networks. Specifically, the diameter of an overlay network affects the number of hops, i.e., a TTL value, required to find a sufficient number of provider peers and files desired by users. Furthermore, if an overlay network is constructed without taking into account the topology of underlying physical networks, a logical link may be established between physically-distant peers. Passing a message from one peer to another takes time and much network resources. A peer has to wait for a long time to obtain information on physically-close providers from which it can quickly retrieve its desired file.

To tackle these problems, an overlay network should reflect the characteristics of the underlying physical topology, e.g., the degree distribution and the physical proximity. In this thesis, we propose novel methods to construct a low-diameter and location-aware overlay network where peers can find physically-close providers without introducing much load on underlying physical networks [55]. There have been several research works on the construction of a low-diameter network. Barabási-Albert (BA) model [56] is the first and the most popular model that explains generation of a network whose degree distribution follows the power-law. The BA model is dominated by the preferential attachment (PA). The PA defines the probability P_i that node i gets links from newly added nodes as proportional to its degree k_i, that is $P_i = \frac{k_i}{\sum_{j \in S_n} k_j}$, where S_n is the set of nodes existing in the network. As a result, it constructs a scale-free network in which there are a few hub nodes that are connected with many low-degree nodes. This characteristic contributes to reduction of the diameter of the network, where a peer can discover a sufficient number of desired files in the range of a small number of logical hops.

Although the BA can build an overlay network, it does not consider the topology of underlying physical networks. We should shorten the physical distance, i.e., the

number of physical hops, between neighboring peers to reduce the load on underlying physical networks. For this purpose, in our model a peer adopts the PA to only peers that are physically close. As a result, constructed overlay network has a correlation between the logical and the physical distances. Usually in conventional P2P file-sharing systems, on receiving a response message, a peer has to examine the physical closeness of the peer by using a ping message. On the other hand, in our overlay network, logically-close peers are also physically close. It does not require for a peer to probe the physical distance from the discovered providers by emitting additional ping-like messages.

Since a new peer only knows some of peers notified by a bootstrapping node in realistic situations, it can not necessarily find appropriate peers from the initially obtained peer list and it leads to the limitation of the effectiveness of a constructed network in join phases. To improve the structure of a constructed network, a rewiring method is viable, which enables a peer to connect to more appropriate peers after the join phase [57-59]. In this thesis, we also propose a new rewiring method to improve the location-awareness of an overlay network. We assume that a peer can obtain information on peers that its neighboring peers know by periodically exchanging ping-pong messages. Based on the constructed peer list, a peer finds peers that are physically closer and with a higher-degree than the current neighbors.

The rest of this thesis is organized as follows. Chapter 2 explains proxy caching mechanisms with a media quality adjustment. After introducing the overview of our system, we propose three mechanisms: media retrieval mechanism, media prefetching mechanism, and cache replacement mechanism. Through several simulation experiments, we evaluate our proposed mechanisms in terms of required cache buffer size, play-out delay, and media quality. Simulation results show that our system is effective in suppressing the play-out delay and reducing the required cache buffer size with a media stream of the desired quality. We further implement our proposed mechanisms

on a real system. Through experiments, it is shown that our implemented system can continuously provide users with a high-quality and low-delay media service in accordance with the network condition. Chapter 3 describes continuous and scalable media streaming on unstructured P2P networks. We first propose scalable media search methods taking into account temporal order of reference to media data that means users usually play out a media stream from the beginning to the end. Then, we also propose media retrieval methods to accomplish continuous media play-out. Through several simulation experiments, we show that the proposed methods can provide users with continuous media play-out without introducing extra load on the system. Chapter 4 discusses adaptive and robust P2P media streaming. First, we propose a novel cache replacement algorithm that considers balance between supply and demand for media streams. Next, to improve the adaptability and robustness against changes of system conditions, such as network conditions, and peer departures, we further propose an adaptive media retrieval method and a robust media search method. Simulation results show that our proposed cache replacement algorithm can accomplish continuous media play-out independent of media popularity and adapt to changes in media popularity. Furthermore, we demonstrate that the adaptive media retrieval method can provide users with more continuous media play-out than the previous method proposed in Chapter 3. In addition, we find that the proposed search method is robust to peer departures. Chapter 5 focuses on the construction of a P2P overlay network to improve the user QoS in terms of search and retrieval time of files. After introducing several related works, we propose a construction method of a low-diameter and location-aware P2P overlay network where a peer can find many physically-close provider peers. We further propose a rewiring method to improve the structure of the constructed overlay network. Through evaluation in terms of diameter, physical distance between neighboring peers, and degree distribution of the overlay network, it is shown that our proposed method without rewiring

can construct a low-diameter overlay network comparable with the BA model. In addition, we find that the rewiring method contributes to reduction of both the physical distance between neighboring peers and diameter of the overlay network. Finally, Chapter 6 concludes this thesis and presents future research works.

Chapter 2

Proxy Caching Mechanisms with Quality Adjustment for Media Streaming Services

2.1 Introduction

With the growth of computing power and the proliferation of the Internet, media streaming services have widely diffused. Then, a considerable amount of media traffic injected by the services causes serious congestion and, as a result, network cannot provide users with the high-quality and interactive services.

The proxy mechanism widely used in WWW systems offers low-delay delivery of data by means of "proxy server". A proxy server caches media data which have passed through it in its local buffer, called "cache buffer", then it provides the cached data to users on demand. By applying proxy mechanisms to a media streaming system, high-quality and low-delay media distribution can be accomplished without introducing extra load on the system [13-20]. In addition, it is effective to adapt the

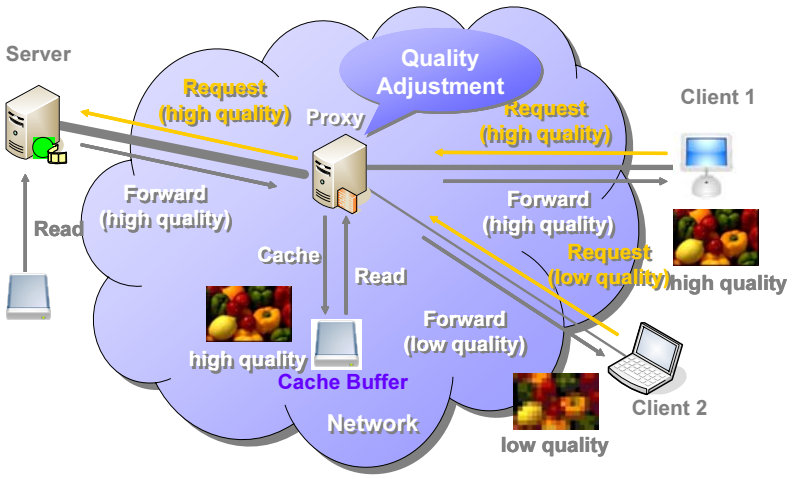


Figure 2.1: Media streaming system

quality of cached media data in the proxy if user requests are different and diverse due to heterogeneity in terms of the available bandwidth, end-system performance, and user's preferences on the perceived media quality [13]. Taking into account the heterogeneity among clients is indispensable when we want to provide users with a distributed multimedia service of a satisfactory level of quality.

In this chapter, we propose proxy caching mechanisms to accomplish high-quality and low-delay media streaming services in a heterogeneous environment. In our proposed system illustrated in Fig. 2.1, a media stream is divided into blocks for efficient use of a cache buffer. A proxy cache server is assumed to be able to adjust the quality of cached or retrieved media blocks to requests through media filters or transcoders. We develop effective algorithms for determining the quality of a block to retrieve from a media server, replacing cached blocks with a newly retrieved block, and prefetching blocks in prior to requests. These algorithms are important when we want to suppress the possibility of cache misses and decrease the block transfer delay introduced by retrieving missing blocks from a distant media server.

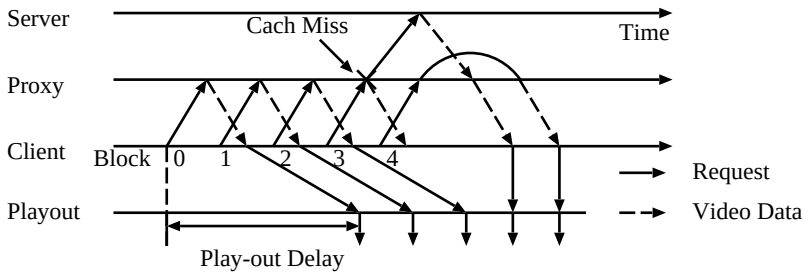


Figure 2.2: Basic behavior of our mechanisms

Through simulation experiments, it is shown that our system with the above algorithms can provide users with high-quality and low-delay media streaming services. Furthermore, we implement proposed mechanisms on a real system to verify their practicality and usefulness. We conduct several experiments and evaluate the traffic condition, the media quality variation, and the overhead of quality adjustment. Then, we confirm that our implemented system can continuously provide users with a media stream in accordance with the network condition.

This chapter is organized as follows. In section 2.2, we propose mechanisms for a proxy cache server with a media-quality adjustment mechanism and we evaluate our proposed mechanisms through several simulation experiments. In section 2.3, we describe implementation of proposed mechanisms on a real system and conduct several experiments to evaluate our proposed mechanisms. Finally, we conclude the results of our study in section 2.4.

2.2 Proxy Caching Mechanisms with Media-Quality Adjustment

2.2.1 Overview of Mechanisms

Figure 2.2 illustrates the basic behavior of our mechanisms. Considering the reusability of cached data, a media stream is divided into N blocks [13-15].

A client periodically requests a designated proxy to send a block. Each request expresses the desired level of quality of the block, which is determined based on the client-system performance, user's preferences on the perceived media quality, and the available bandwidth specified by an underlying protocol, e.g., TFRC (TCP Friendly Rate Control) [7] or the link capacity.

The proxy maintains a table, called "cache table", for each of media streams and possesses informations on cached blocks. Each entry includes the block number, the size and quality of the cached block and the quality of blocks under transmission. The size and quality become zero if the block is not cached or not being transmitted.

On receiving a request, a proxy compares it to a corresponding entry in the table. If the quality of a cached block can satisfy the request, i.e., cache hit, the proxy reads out the cached block, adjusts the level of the quality to the request, and transmits it to the client. Media-quality adjustment is performed by QoS filtering techniques such as frame dropping, low-pass, and re-quantization filters [21]. In some cases, a block being received can satisfy the request. To avoid introducing extra delay in retrieving a block, the proxy waits for the completion of the reception and provides it to the client. Otherwise, when a cache misses the request, the proxy retrieves a block of the appropriately determined quality from the server on a session established to the server for the client. Then, the newly obtained block is stored in the cache. If there is not enough room, one or more cached blocks are replaced with the new block. Cached

blocks are useful to reduce the response time, but further reduction can be expected if the proxy retrieves and prepares blocks in advance using the residual bandwidth.

In the following subsections, we propose several algorithms for the block retrieval, prefetching, and replacement mechanisms.

2.2.2 Block Retrieval Mechanism

When a cache cannot supply a client with a block of the requested quality, a proxy should retrieve the block. The quality of a block that a proxy can retrieve from a server in time is determined in accordance with the available bandwidth between the server and the proxy. Thus, when the path between the server and the proxy is congested, the proxy cannot satisfy the client's demand even if it retrieves the block from the server. We introduce a parameter α which is given as;

$$\alpha = \frac{\max(Q_{sp}(i,j), Q_{cache}(j))}{Q_{pc}(i,j)}. \tag{2.1}$$

α is the ratio of the quality that the proxy can provide to the request. j $(1 \leq j \leq N)$ is the block number that client i requires. $Q_{sp}(i,j)$ stands for the quality of block j that can be transfered from the server to the proxy within a block time. The block-time is given by dividing the number of frames in a block by the frame rate. For example, in our evaluations, the block corresponds to 30 frames and it is played out at 30 frames per second. Thus, one block-time is equal to one second. By multiplying the available bandwidth by the block-time, the proxy obtains the size feasible for block j. Then, assuming that the quality can be determined from the size [60], we can derive $Q_{sp}(i,j)$. The quality affordable on the path between the proxy and the client is expressed as $Q_{pc}(i,j)$ and is regarded as client i's request on block j. The quality of a cached block j, $Q_{cache}(j)$, is obtained from a corresponding entry of the cache table. In this subsection, since we consider a case of the cache miss, $Q_{cache}(j) < Q_{pc}(i,j)$

holds.

When $\alpha \geq 1$, that is, the proxy can provide the client with the block of the desired quality, we have three alternatives of determining the quality of block j to retrieve for client i, $Q_{req}(i,j)$.

method1: A possible greedy way is to request the server to send block j of as high quality as possible. This strategy seems reasonable because cached blocks can satisfy the most of the future requests and probability of cache misses becomes small. Then, the request $Q_{req}(i,j)$ becomes

$$Q_{req}(i,j) = Q_{sp}(i,j). \tag{2.2}$$

method2: When the available bandwidth between the server and the proxy is extremely larger than that between the proxy and the client, method1 cannot accomplish the effective use of bandwidth and cache buffer. Thus, we propose an alternative which determines the quality $Q_{req}(i,j)$ based on prediction of demands on block j, as follows;

$$Q_{req}(i,j) = \min(\max_{k \in S, 0 \leq l \leq j} Q_{pc}(k,l), Q_{sp}(i,j)), \tag{2.3}$$

where S is a set of clients which are going to require block j in the future. Client i is also in S.

method3: To accomplish a further efficient use of the cache, it is possible to request block j of the same quality that the client requests, as follows;

$$Q_{req}(i,j) = Q_{pc}(i,j). \tag{2.4}$$

With this strategy, the number of cached blocks increases and the probability of cache misses is expected to be suppressed as far as future requests can be satisfied with

them.

In some cases, both cached and retrieved blocks cannot meet the demand ($\alpha < 1$). One way is to request a server to send a block of the desired quality, but it may cause undesirable delay. The other is for a client to be tolerant of the quality degradation and accept a block whose quality is lower than the request. We introduce another parameter β to tackle this problem. β is defined as the ratio of the acceptable quality to the demand, and it expresses the client's insistence on the media quality. Clients with β close to one want to receive blocks in accordance with the request at the risk of undesirable transfer delay. On the other hand, those who value timeliness and interactivity of applications will choose β close to zero.

First, we consider the case that the quality of a cached block can satisfy the client, but is still lower than the request ($\beta \leq \frac{Q_{cache}(j)}{Q_{pc}(i,j)} \leq \alpha < 1$). In such a case, in order to effectively reuse the cached block, the proxy only sends the cached block to the client regardless of the quality of the block that the server can provide, $Q_{sp}(i,j)$. When the quality of the cached block is not high enough ($\frac{Q_{cache}(j)}{Q_{pc}(i,j)} < \beta \leq \frac{Q_{sp}(i,j)}{Q_{pc}(i,j)} = \alpha < 1$), the proxy requests the server to send block j whose quality is equal to $Q_{sp}(i,j)$ as;

$$Q_{req}(i,j) = Q_{sp}(i,j). \tag{2.5}$$

Finally, if the proxy cannot provide the client with a block of the satisfactory quality ($\alpha < \beta$), it requests the server to send the block of the minimum quality which is expected not to cause a cache miss, that is,

$$Q_{req}(i,j) = \beta \cdot Q_{pc}(i,j). \tag{2.6}$$

The proxy requests the server to send block j of the quality $Q_{req}(i,j)$. The corresponding entry for client i $Q_{rec}(i,j)$ for the quality of block j being received in

the cache table is set to $Q_{req}(i,j)$. When the block reception is finished, $Q_{cache}(j)$ is set to $Q_{rec}(i,j)$.

2.2.3 Block Prefetching Mechanism

To reduce the possibility of cache misses and avoid the delay in obtaining missing blocks from a server, a proxy prefetches blocks that clients are going to require in the future. After checking the cache table for block j being requested by client i, a proxy compares the minimum requirement $\beta \cdot Q_{pc}(i,j)$ to the quality of cached blocks $Q_{cache}(k)$ and that of receiving blocks $Q_{rec}(i,k)$ (for $\forall i,\ j+1 \leq k \leq j+P \leq N$). Here, P is the size of a sliding window called a prefetching window, which determines the range of examination for prefetching. If there exists any block whose quality is lower than the minimum, a block retrieval mechanism is triggered. The mechanism is the same as one explained in subsection 2.2.2 except that the available bandwidth to prefetching is the remaining bandwidth between the server and the proxy. Prefetch requests have a lower priority than requests for retrieving cache-missed blocks at the server in order not to disturb urgent block-retrieval.

2.2.4 Cache Replacement Mechanism

Since a cache buffer has a limited capacity, the replacement of cached blocks should be considered to accomplish the effective use of a storage. When the quality of a newly retrieved block is lower than that of a cached block, the new block is not to be cached. Otherwise, a proxy first removes a cached block of the lower quality if exists. Then, it tries to deposit the new block in the cache. If there is not sufficient room to store the new block, some cached blocks must be removed. We propose a replacement algorithm with consideration of size, quality, and re-usability of blocks.

First, the proxy assigns priority to cached blocks. Blocks requested by clients at

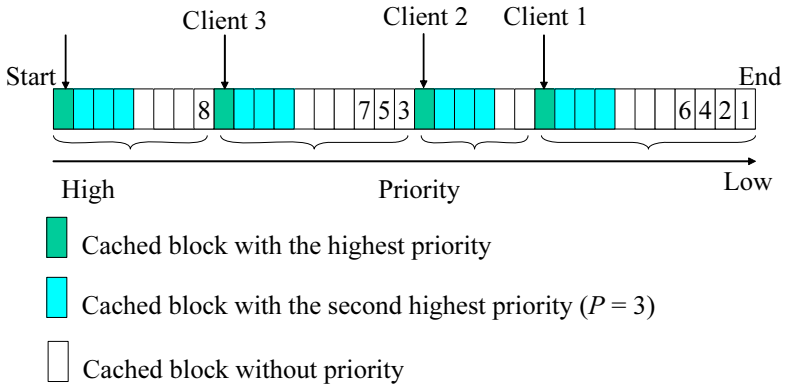


Figure 2.3: Priorities of cached blocks

the moment of the replacement have the highest priority and are never removed from the cache. The block residing at the beginning of the stream is also assigned the highest priority to provide potential clients with a low-latency service. The second important blocks are those in the prefetching windows following the most important blocks. The other blocks are with no priority.

Then, blocks candidate for replacement are chosen one by one until the sufficient capacity becomes available. In Fig. 2.3, we show an example of victim selection. A cached block, which locates at the end of longest succession of un-prioritized blocks, is regarded as the least important and becomes the first victim as indicated as "1" in the figure. Among successions of the same length, one closer to the end of the stream has a lower priority.

The proxy first tries the quality adjustment to decrease the size of the victim if it is larger than the new block. Since it is meaningless to hold a block whose quality is smaller than $Q_{req}(i, j)$ determined by the method chosen in the block retrieval mechanism, no further adjustment is performed and the victim is removed from the cache. The proxy repeatedly chooses the next victim and applies the same techniques until the capacity for the new block becomes sufficient. When all un-prioritized blocks

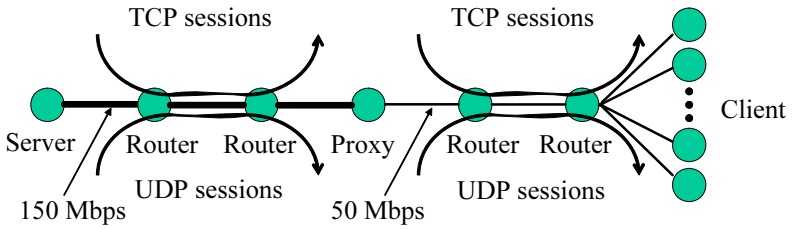


Figure 2.4: Simulation system model

are removed but it is still insufficient, the proxy gives up storing the new block.

2.2.5 Simulation Experiments

In this section, we conduct simulation experiments to evaluate performance of the proposed caching mechanisms in terms of the required buffer size, the playout delay, and the media quality.

Figure 2.4 illustrates our simulation system model. A media stream of two hours long is coded using the MPEG-2 media coding algorithm. It is segmented into GoPs (Group of Pictures) and a GoP corresponds to a one-second block. The media-quality adjustment is performed by a re-quantization filter which regulates the quantizer scale, that is, the degree of the quantization. The size of entire media stream ranges from 8.6 Gbits to 194.5 Gbits according to the applied quantizer scale. Ten clients are connected to the proxy on the same path and watch the same media stream from the beginning to the end without interactions such as rewinding, pausing, and fast-forwarding. The inter-arrival time between two successive client participations follows the exponential distribution whose average is 1,800 seconds. The propagation delay between the server and the proxy is 200 msec and that between the proxy and the client is 50 msec. The simulation runs for 29,000 seconds in simulation time unit. It is assumed that all sessions employ TFRC as an underlying rate-control protocol. We conduct simulation experiments on TFRC with ns-2 [61] and obtained results are

used as the available bandwidth of sessions.

Although requests are sent to the proxy at the regular interval of a block-time, inter arrival times of blocks at the client fluctuate due to cache-hit, cache-miss, and available bandwidth. In any types of streaming services, it is necessary for a client to defer the playout and buffer some amount of media preparing for expected or unexpected delay jitter (See Fig. 2.2). We define the time that client i waits and buffers media blocks at the beginning of the session in order to ensure regularity and smoothness of media playout as the playout delay $W(i)$. $W(i)$ is derived as;

$$W(i) = \max_{1 \le j \le N} (T(i,j) - I(i,j)), \tag{2.7}$$

where j stands for the block number, and N is the number of blocks in the stream. The arrival time of block j at client i is denoted as $T(i,j)$. $I(i,j)$ corresponds to the ideal arrival time of block j and those conditions hold that $I(i,j) - I(i,j-1) =$ one block-time and $I(i,1) = T(i,1)$.

Next, we define the degree of user's satisfaction with media quality as;

$$S(i) = \frac{1}{N} \sum_{j=1}^{N} \frac{Q_{act}(i,j)}{Q_{pc}(i,j)}, \tag{2.8}$$

where $Q_{act}(i,j)$ is the quality of block j provided to client i whose request on the block is $Q_{pc}(i,j)$.

In Figs. 2.5 through 2.7, we summarize simulation results on the amount of cached blocks and the playout delay. The proxy is assumed to have an infinite cache buffer. Prefetching window size is set to zero, i.e., no prefetching, in Figs. 2.5 and 2.6, and 30 in Fig. 2.7. Client's insistence on the quality β is set to 1. Those results labeled "traditional" correspond to the case that the proxy does not have capability of neither quality adjustment or prefetching. In such a case, a cache hits a request

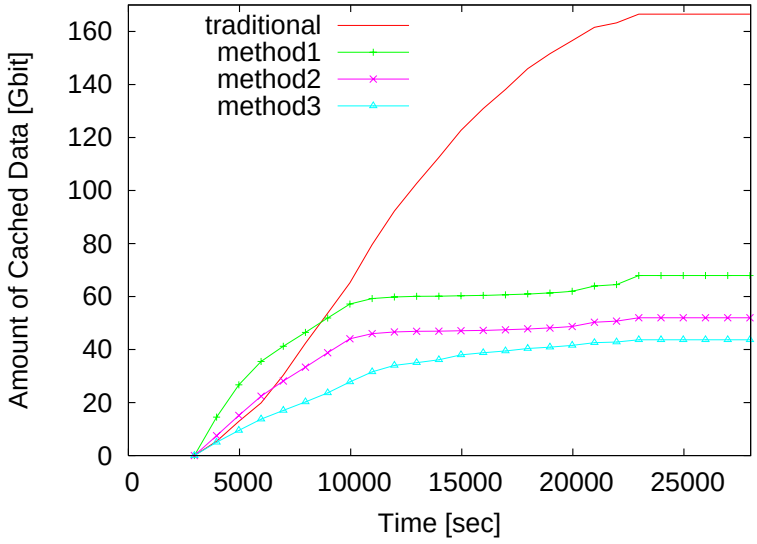


Figure 2.5: Amount of cached data with infinite cache ($\beta = 1$, $P = 0$)

only when it has a media block of the same quality as the request. Otherwise, the proxy retrieves the block of the requested quality from the server. The traditional proxy tries to store every blocks it retrieves even if it already has a block of the same number and the higher quality. Figure 2.5 shows that the required buffer size of proposed methods is down to one forth of the traditional method while providing clients with media blocks of the requested quality. In addition, even if clients insist on the quality, the playout delay is suppressed by introducing the quality adjustment and the prefetching mechanism as shown in Figs. 2.6 and 2.7. In the case of method3, the prefetching mechanism is not so effective in comparison with the others because the proxy retrieves and caches blocks of the minimum quality.

Next, we show simulation results for the case where the proxy is equipped with the cache of 20 Gbits, which is smaller than the half of that required (see Fig. 2.5). Since we cannot expect an efficient use of cached blocks with obstinate clients, we assume that they are tolerant of quality degradation, that is, $\beta = 0.6$.

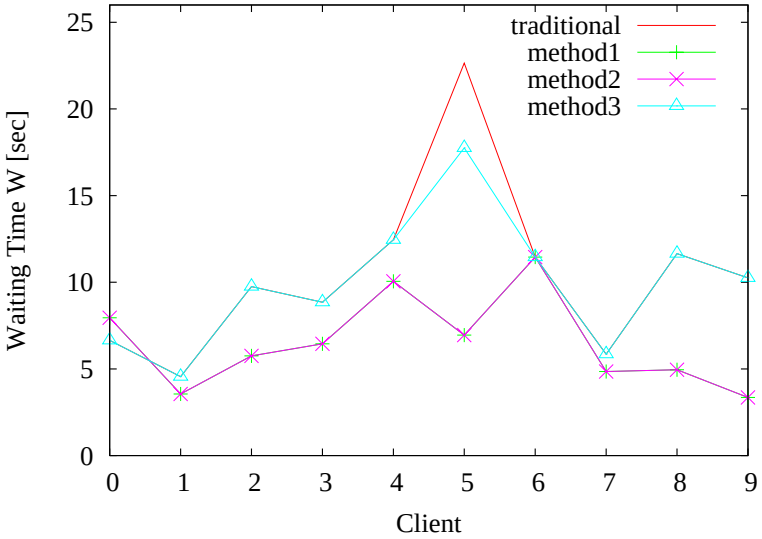


Figure 2.6: Playout delay with infinite cache ($\beta = 1$, $P = 0$)

The lower β leads to the higher cache-hit probability. Consequently, regardless of methods, the playout delay becomes small enough while the amount of cached blocks is limited to 20 Gbits as shown in Fig. 2.8. Furthermore, since method3 requests the server to send blocks of the lowest quality among three methods, the block transfer time in method3 becomes small and the number of cached blocks increases. As a result, performance improvement of method3 becomes high. Due to a limited cache buffer, the degree of satisfaction $S(i)$ slightly decreases but is still higher than 0.6 as shown in Fig. 2.9.

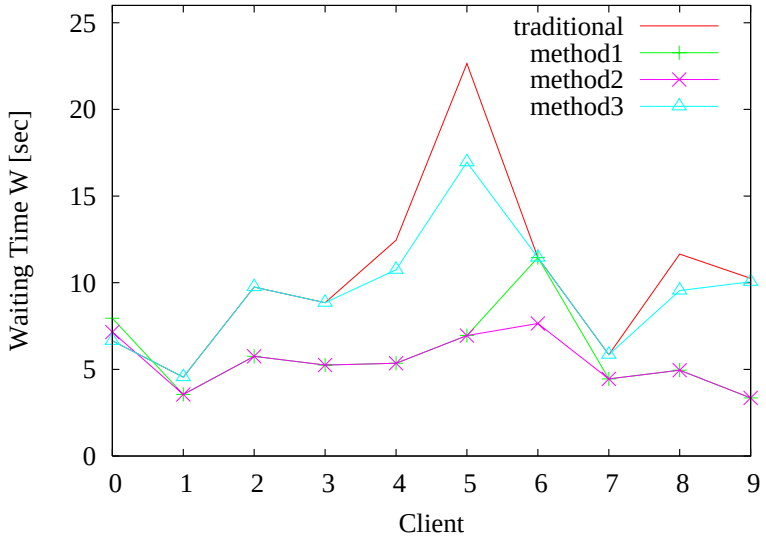


Figure 2.7: Playout delay with infinite cache ($\beta = 1$, $P = 30$)

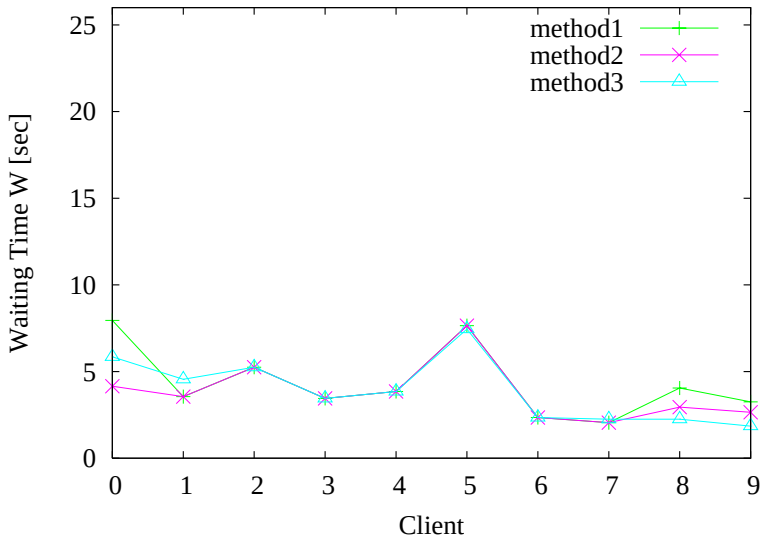


Figure 2.8: Playout delay with 20 Gbits cache ($\beta = 0.6$, $P = 30$)

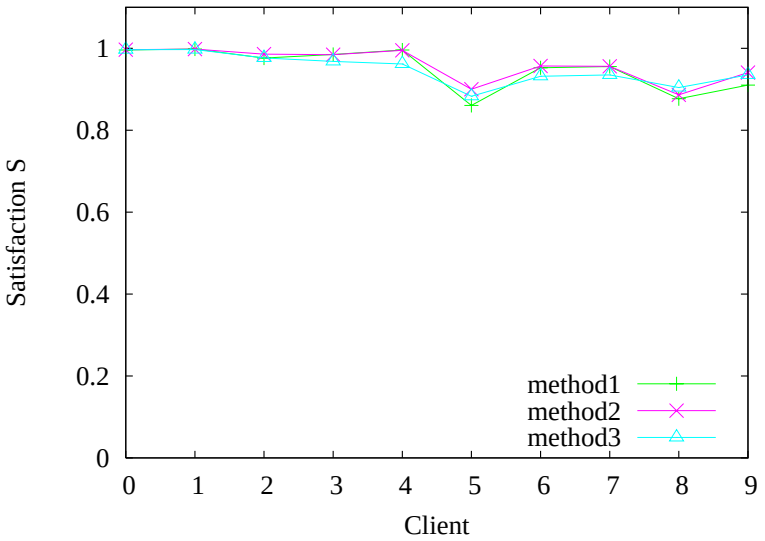


Figure 2.9: Degree of satisfaction on delivered media with 20 Gbits cache ($\beta = 0.6$, $P = 30$)

2.3 Implementation of Proposed Mechanisms

In this section, we describe our implementation of proposed mechanisms on a real system. Figure 2.10 illustrates modules constituting our media streaming system. The implemented system consists of a media server, a proxy cache server, and several clients. We employ well-known and widely-used protocols for inter-system communications. For example, the media streaming is controlled through RTSP (Real Time Streaming Protocol) / TCP sessions. Media blocks are transferred over RTP (Realtime Transport Protocol) / UDP sessions as being segmented into 1 K bytes-long RTP packets. The media stream is coded using the MPEG-2 media coding algorithm in the PS (Program Stream) format. The available bandwidth, that is taken into account in three mechanisms explained in section 2.2, is determined by TFRC. The media-quality adjustment is performed by a low-pass filter [21]. In the following

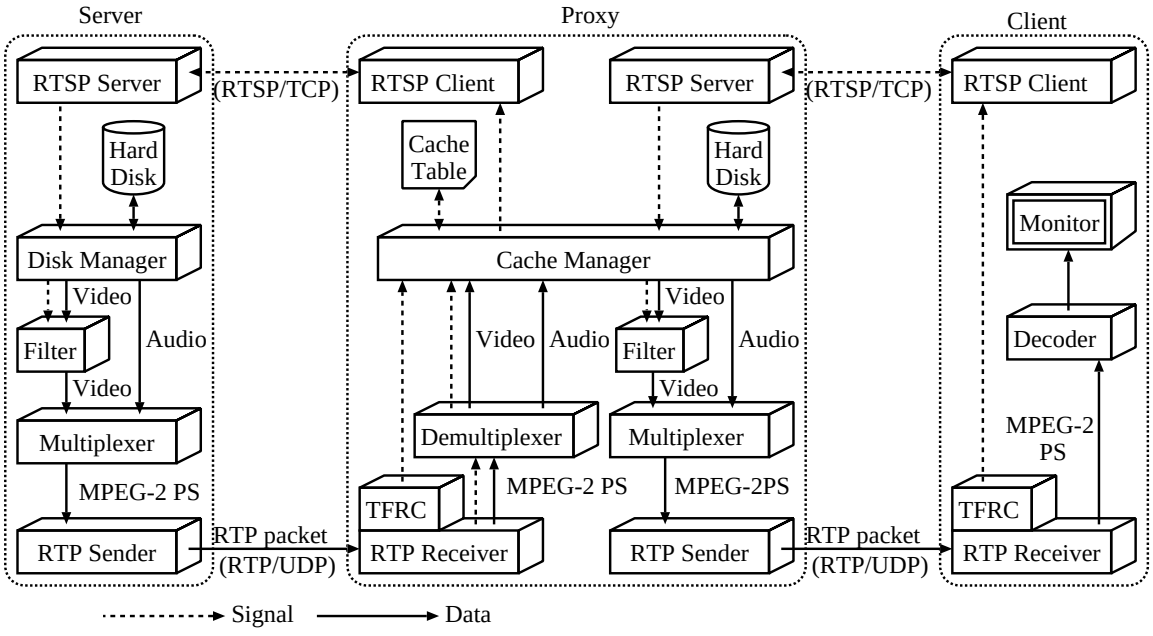


Figure 2.10: Modules constituting system

subsections, details of our implementation are given.

2.3.1 Delmultiplexing MPEG-2 PS Blocks

MPEG-2 PS is one of the formats for multiplexing media and audio streams. As the quality adjustment is applied only to media data, a block received through a proxy's *RTP Receiver* is divided into a pair of media and audio blocks by *Demultiplexer*. The divided blocks are stored in a cache buffer separately. In our implemented system, each block corresponds to a GoP (Group of Pictures) of MPEG-2, which consists of a series of frames.

The block to request and its quality are specified in the header of an RTSP PLAY message using the Range field and Bandwidth field, respectively. In a case of a cache hit, a proxy reads out both media and audio blocks from its cache, and it applies the quality adjustment only to the media block. Then, those blocks are multiplexed and transmitted to the requesting client.

2.3.2 Rate Control with TFRC

TFRC is a protocol that enables a non-TCP session to behave in a TCP-friendly fashion [7]. TFRC sender estimates the throughput of a TCP session sharing the same path in accordance with network condition, expressed in terms of the packet loss probability and RTT. Those informations are obtained by exchanging RTCP (Real-Time Control Protocol) messages between a sender and a receiver. In our implemented system, we can use RTSP as a feedback mechanism where a client calculates the TCP-friendly rate and informs a proxy of the rate using the Bandwidth field of a RTSP PLAY message.

2.3.3 Media-Quality Adjustment

We employ the low-pass filter as a quality adjustment mechanism. We compared several media filters such as the low-pass, re-quantization, and frame dropping [22]. Through experiments, it was shown that the low-pass filter is the most suitable as an MPEG-2 media filter for its flexibility in rate adaptation, faster processing, and media quality. The low-pass filter adjusts the media quality to the desired level by discarding some portion of less influential information in media blocks.

2.3.4 Cache Manager

A proxy maintains information on cached blocks as *Cache Table*. On receiving a request, *Cache Manager* examines the table. When a cache miss occurs, it determines the quality of a block to retrieve from a media server in accordance with the available bandwidth and requests using methods explained in section 2.2. Then, it requests the server to send the block via an RTSP session established between them.

The server reads out a pair of a media block of the highest quality and a corresponding audio block through *Disk Manager*, adjusts the quality of the media block to the request using *Filter*, rebuilds a PS block by *Multiplexer*, and finally sends the block to the proxy via an RTP session in a TCP-friendly fashion.

Cache Manager obtains a pair of blocks through *RTP Receiver* and *Demultiplexer*. The block is sent to the client in a similar way to the block transfer from the media server to the proxy. At the same time, a pair of blocks is stored in *Hard Disk*. *TFRC* calculates TCP-friendly rate while receiving RTP packets from the server.

Cache Manager is also responsible for prefetching blocks and replacing cached blocks with new blocks.

2.3.5 Experimental Evaluation

In this section, we conduct experiments to evaluate the rate variation, the media quality variation, and the overhead of quality adjustment.

Figure 2.11 illustrates a configuration of our experimental system. Two clients are connected to the proxy and watch the same media stream of 10 minutes from the beginning to the end without interactions such as rewinding, pausing, and fast-forwarding. The media server has the whole media blocks of the highest quality of 8 Mbps. The proxy also has the whole media blocks, but the quality is 3 Mbps and a cache buffer capacity is limited to 450 MBytes. The proxy determines the quality of a block to retrieve from a media server using method3 which does not need to adjust the quality of a newly retrieved block at the proxy. The prefetching window size P is set to 5. There exists a TCP session for the file-transfer as a disturbance that competes with media sessions for the bandwidth. The client's insistence on the media quality, β, is set to 1. Clients defer playing the media until media data of 4 MBytes are stored in playout buffer.

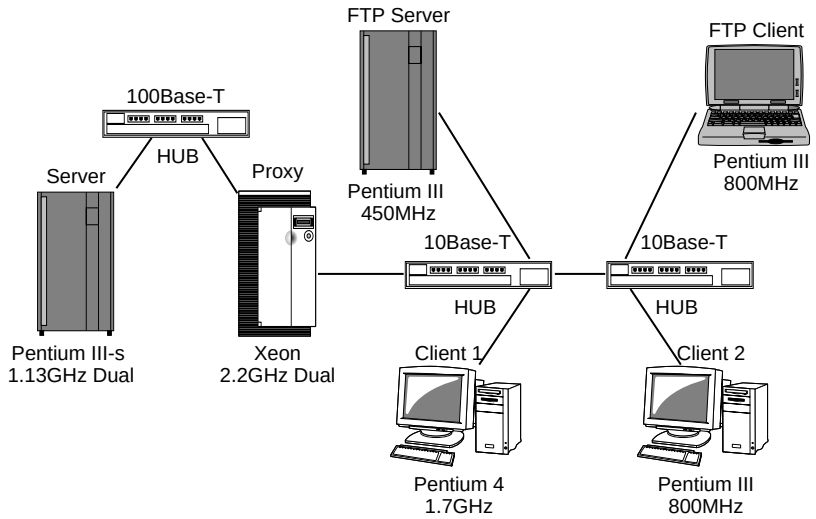


Figure 2.11: Configuration of experimental system

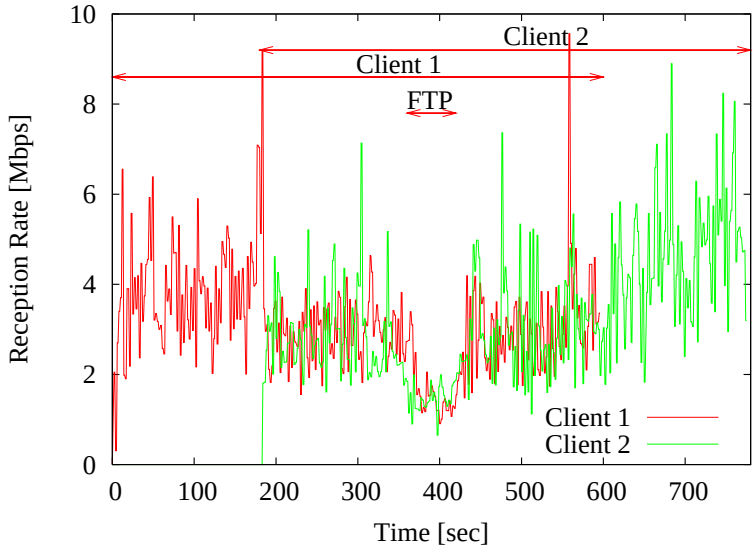


Figure 2.12: Reception rate variation

Figure 2.12 illustrates variations in reception rates observed at the clients' *RTP Receiver*. At time 180, the client 2 begins a media session. From 360 to 420, the TCP session transfers a file. As shown in the figure, the media rate is regulated as the network condition changes, to avoid unexpected transfer delay and quality degradation that would be caused by congestions.

Figure 2.13 illustrates variations in the perceived media quality in terms of the coding artifact measured by VP2000A of KDD Media Will Corporation. A higher coding artifact value means that quality degradation is higher. These figures show that the perceived media quality changes in accordance with the network condition.

Through experiments, the maximum processing time of adjusting the quality of a media block was less than 0.5 second. Since the proxy is required to process each block faster than one second, i.e., the interval between two consecutive requests in our experiments, more than 0.5 second can be devoted to the other tasks including retrieving a block from the media server. In our experiments, we observed only

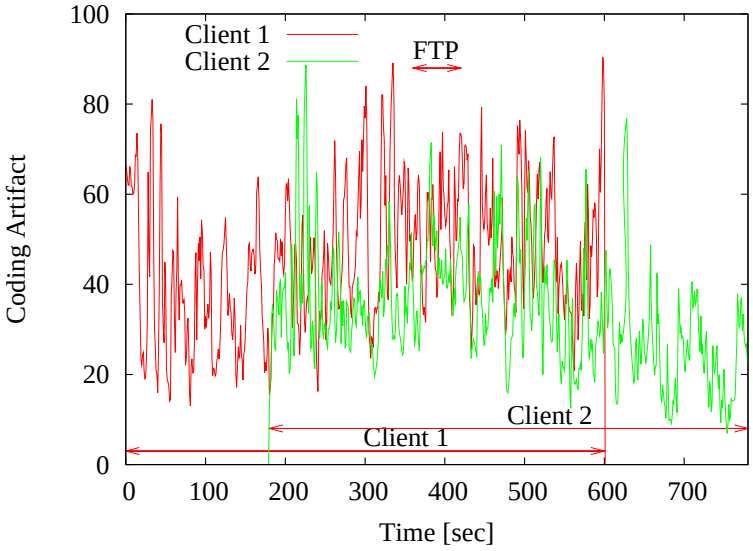


Figure 2.13: Media quality variation

several freezes during 10 minutes media playout. Thus, we can conclude that our mechanisms can accomplish a low-latency and high-quality media streaming service under a heterogeneous and dynamically changing environment.

2.4 Conclusion

In this chapter, we proposed several caching mechanisms for the media streaming system with the proxy server capable of media-quality adjustment. Simulation results show that our system is effective enough in suppressing the playout delay and reducing the required cache size while providing users with a media stream of the desired quality. Specifically, waiting time for media play-out is less than 8 seconds even under a severe situation where the cache buffer size is limited to 20 Gbits. Furthermore, we implemented and evaluated our proposed mechanisms on a real system. Through the experiments, it is shown that our implemented system can continuously provide

users with a high-quality and low-delay media service in accordance with the network condition.

Chapter 3

Continuous and Scalable Media Streaming on P2P Networks

3.1 Introduction

With the growth of computing power and the proliferation of broadband access to the Internet, such as ADSL and FTTH, the use of media streaming has become widely diffused. A user receives a media stream from an original media server through the Internet and plays it out on his/her client system as it progressively arrives. However, with the current Internet, the major transport mechanism is still only the best effort service, which offers no guarantees of bandwidth, delay, and packet loss probability. Consequently, such a media streaming system cannot provide users with media streams in a continuous way.

P2P is a new network paradigm in which hosts called peers directly communicate with each other and exchange information without the mediation of servers. Such distributed manner improves the system scalability against the increase of the users because traffic concentration to a specific point can be avoided. We expect that

P2P architectures can support media streaming services based on the traditional client-server architectures. There have been several research works on P2P media streaming [26-28, 30, 34, 62]. Most of these have constructed an application-level multicast tree whose root is an original media server while the peers are intermediate nodes and leaves. Their schemes were designed for use in live broadcasting. Thus, they are effective when user demands are simultaneous and concentrated on a specific media stream. However, when demands arise intermittently and peers request a variety of media streams, as in on-demand media streaming services, an efficient distribution tree cannot be constructed. Furthermore, the root of the tree, that is, a media server, can be regarded as a critical point of failure because such systems are based on the client-server architecture.

In Ref. [45], we proposed scalable search and in-time media retrieval methods for on-demand media streaming on unstructured P2P networks. In our system, every peer participating in a service watches a media stream and deposits it in its local cache buffer. A media stream is divided into blocks for efficient use of network bandwidth and cache buffer [15, 16, 18, 63]. By retrieving blocks from other peers in time, a peer can watch a desired media stream. Since there is no server that manages information on peer and media locations, a peer has to find each block constituting a desired media stream by emitting a query message into the network. Other peers in the network reply to the query with a response message and relay the query to the neighboring peers. If a peer successfully finds a block cached in other peers, it retrieves it from one of them and deposits it in its local cache buffer. If there is no room to store the newly retrieved block, a peer has to perform replacement on cached blocks with it.

There are several issues to resolve in accomplishing effective media streaming over unstructured P2P networks. Scalability is the most important among them. Flooding, in which a peer relays a query to every neighboring peer, is a powerful

scheme for finding a desired media stream. However, it has been pointed out that the flooding lacks scalability because the number of queries that a peer receives significantly increases with the growth in the number of peers [51]. In particular, a block-by-block search by flooding apparently introduces much load on the network and causes congestion. To tackle this problem, we proposed two scalable block search methods. Taking into account the temporal order of reference to media blocks, a peer sends a query message for a group of consecutive blocks. Then, the peer performs adaptive block search by regulating the search range based on the preceding search result.

Since continuous media play-out is the most important factor for users in media streaming services, we have to consider a deadline of retrieval for each block. To retrieve a block by its corresponding play-out time, we proposed methods to determine an appropriate provider peer (i.e., a peer having a cached block) from search results by taking into account the network conditions, such as the available bandwidth and the transfer delay. By retrieving a block as fast as possible, the remaining time can be used to retrieve the succeeding blocks from distant peers.

Through several simulation experiments, we compare several combinations of those methods and algorithms, in terms of the amount of the search traffic and the continuity of the media play-out.

The rest of this chapter is organized as follows. In Section 3.2, we give an overview of our streaming system on P2P networks and describe our per-group based search and retrieval methods. Next, in Section 3.3, we evaluate our proposed methods through several simulation experiments. Finally, we conclude the results of our study in Section 3.4.

3.2 Search and Retrieval Methods for Media Streaming on P2P Networks

Figure 3.1 illustrates our media streaming system on unstructured P2P networks. A peer participating in our system first joins a logical P2P network for the media streaming. Then, a peer sends a query message to find a media stream that it wants to watch. We especially call this query as a media request. For efficient use of network bandwidth and cache buffer, a media stream is divided into blocks. A peer searches, retrieves, and stores a media stream on a block-by-block basis. In this section, we describe our scalable search methods to find desired blocks and algorithms to determine an appropriate provider peer from the search results.

3.2.1 Per-group based Block Search and Retrieval

In our system, a peer retrieves a media stream and plays it out in a block-by-block manner. However, a block-by-block search apparently increases the number of queries

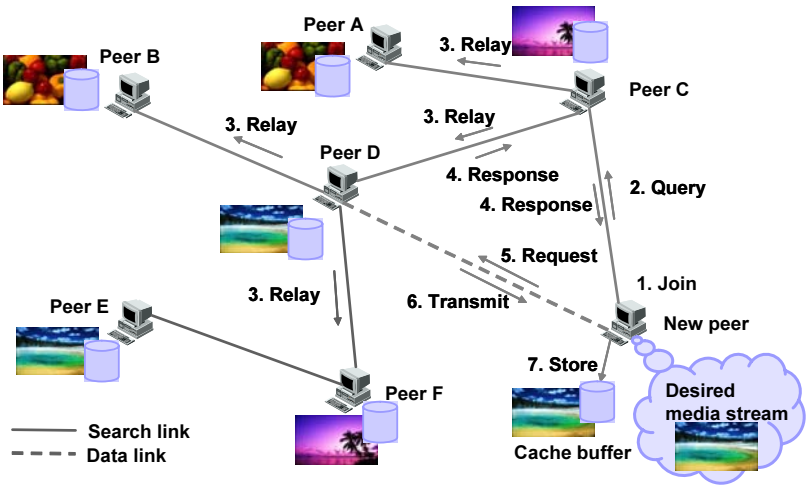


Figure 3.1: Overview of our media streaming on unstructured P2P networks

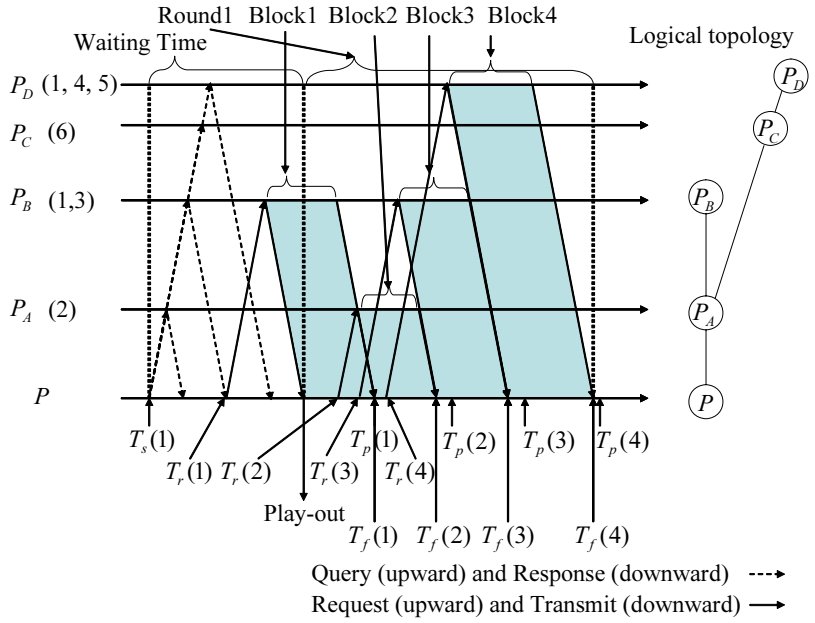


Figure 3.2: Example of per-group search and retrieval

that are transferred on the network and causes network congestion. To tackle this problem, taking into account the temporal order of reference in a media stream, our method employs a per-group search to accomplish scalable media search.

A peer sends out a query message for every N consecutive blocks, called a round. Figure 3.2 illustrates an example of $N = 4$. P_A, P_B, P_C, and P_D indicate peers within the range of the propagation of query messages. Numbers in parentheses next to peers stand for identifiers of the blocks that a peer has. At time $T_s(1)$, a query message for blocks 1 to 4 is sent out from P to the closest peer P_A. The query is relayed among peers. Since P_A, P_B, and P_D have one or more block out of four requested blocks, they return response messages. P determines a provider peer for each block in the round from the search results obtained by the query. It takes two Round Trip Time (RTT) periods from the beginning of the search to the start of reception of the first block of the round. To accomplish continuous media play-out, P sends a query for

the next round at a time that is $2RTT_{worst}$ earlier than the start time of the next round. RTT_{worst} is the RTT to the most distant peer among the peers that returned response messages in the current round.

3.2.2 Scalable Block Search

Since each peer retrieves a media stream sequentially from the beginning to the end, we can expect that a peer that sent back a responses message for the current round has some blocks of the next round. In our methods, a peer tries flooding at the first round. However, in the following rounds, it searches blocks in a scalable way based on the search results of the previous round.

A query message consists of a query identifier, a media identifier, and a pair of block identifiers to specify the range of blocks needed, i.e., $(1, N)$, a time stamp, and Time To Live (TTL). A peer that has any blocks in the specified range sends back a response message. A response message reaches the querying peer through the same path, but in the reversed direction, that the query message traversed. The response message contains a list of all cached blocks, TTL values stored in the received query, and sum of the time stamp in the query and processing time of the query. Each entry of the block list consists of a media identifier, a block number, and block size. If TTL is zero, the query message is discarded. Otherwise, after decreasing the TTL by one, the query message is relayed to neighboring peers except for the one from which it received the query. In the case of Gnutella, a fixed TTL of seven is used. By regulating TTL, the load of finding a file can be reduced. We have called this flooding scheme with a fixed TTL of seven "full flooding," and that with a limited TTL based on the search results, "limited flooding."

In limited flooding, for the kth round, a peer obtains a set R of peers based on response messages obtained at round $k-1$. R is a set of peers expected to have

at least one of the blocks belonging to round k. Since time has passed from the search at round $k-1$, some blocks listed in the response message may have already been replaced by other blocks. Assuming that a peer is watching a media stream without interactions such as rewinding, pausing, and fast-forwarding, and that the cache buffer is filled with blocks, we can estimate the number of removed blocks by dividing the elapsed time from the generation of the response message by one block time B_t. We should note here that we do not take into account blocks cached after a response message is generated. In limited flooding, TTL is set to that of the most distant peer among the peers in R.

To attain an even more efficient search, we also proposed another search scheme. The purpose of flooding schemes is to find peers that do not have any blocks of the current round but do have some blocks of the next round. Flooding also finds peers that have newly joined our system. However, in flooding, the number of queries relayed on the network exponentially increases according to the TTL and the number of neighboring peers [51]. Therefore, when a sufficient number of peers are expected to have blocks in the next round, it is effective for a peer to directly send queries to those peers. We call this "selective search."

By considering the pros and cons of full flooding, limited flooding, and selective search, there are efficient methods based on combining them.

FL method is a combination of full flooding and limited flooding. For blocks of the next round, a peer conducts (1) limited flooding if the conjectured cache contents of peers in R satisfy every block of the next round, or (2) full flooding if one or more blocks cannot be found in the conjectured cache contents of peers in R.

FLS method is a combination of full flooding, limited flooding, and selective search. For the next round's blocks, a peer conducts (1) selective search if the conjectured cache contents of peers in R contain every block of the next round, (2) limited flooding if any one of the next round's blocks cannot be found in the conjectured

cache contents of peers in R, or, finally, (3) full flooding if none of the provider peers it knows is expected to have any block of the next round, i.e., $R = \phi$.

3.2.3 Block Retrieval for Continuous Media Play-out

The peer sends a request message for the first block of a media stream just after receiving a response message from a peer that has the block, because it cannot predict whether any better peer exists at that time. In addition, it is essential for a low-delay and effective media streaming service to begin the media presentation as quickly as possible. Thus, in our method, the peer plays out the first block immediately when its reception starts. Of course, we can also defer the play-out in order to buffer a certain number of blocks in preparation for unexpected delays.

The deadlines for retrieval of succeeding blocks $j \geq 2$ are determined as follows:

$$T_p(j) = T_p(1) + (j-1)B_t, \tag{3.1}$$

where $T_p(1)$ corresponds to the time that the peer finishes playing out the first block.

Although block retrieval should follow a play-out order, the order of request messages does not. We do not wait for completion of reception of the preceding block before issuing a request for the next block because this introduces an extra delay of at least one round-trip, and the cumulative delay affects the timeliness and continuity of media play-out. Instead, the peer sends a request message for block j at $T_r(j)$, which will be given by Equation (3.3), so that it can start receiving block j just after finishing the retrieval of block $j-1$, as shown in Fig. 3.2. As a result, our block retrieval method can maintain the continuity of media play-out.

The peer estimates the available bandwidth and the transfer delay from the provider peer by using existing measurement tools. For example, by using the inline

network measurement technique [64], those estimates can be obtained through exchanging query and response messages without introducing any measurement traffic. Furthermore, the estimates are updated through reception of media data. Every time the peer receives a response message, it derives the estimated completion time of the retrieval of block j, that is $T_f(j)$, from the block size and the estimated bandwidth and delay, for each block to which it has not yet sent a request message. Then, it determines an appropriate peer in accordance with deadline $T_p(j)$ and calculates time $T_r(j)$ at which it sends a request. The detailed algorithm to determine the provider peer is given below.

Step 1 Set j to r, which is the maximum block number among blocks that have already been requested.

Step 2 Calculate set S, a set of peers having block j. If $S = \phi$, that is, there is no candidate provider, set $T_f(j) \leftarrow T_p(j),\ j \leftarrow j + 1$ and repeat Step 2 for the next block. Otherwise, proceed to Step 3.

Step 3 Derive set S', a set of peers from which a peer can retrieve block j by deadline $T_p(j)$, from S. Time required to retrieve block j from provider peer i becomes the sum of round trip time $R(i)$ to peer i and the transfer time of block j obtained by dividing block size $B(j)$ by available bandwidth $A(i)$ from peer i. For each peer i in S, the estimated completion time of the retrieval of block j from peer i is derived as $\max(T_f(j-1), T_{now} + R(i)) + \frac{B(j)}{A(i)}$, considering the case that the retrieval of block $j-1$ lasts more than $R(i)$ and the request for block j is deferred. Here, T_{now} is the time when this algorithm is performed. If the estimated completion time is earlier than $T_p(j)$, the peer is put in S'. If $S' = \phi$, set $T_f(j) \leftarrow T_p(j),\ j \leftarrow j + 1$ and go back to Step 2.

Step 4 Determine provider peer $P(j)$ of block j from S'. We propose the following

two alternative methods for determining the provider peer.

SF (Select Fastest) Method selects a peer whose estimated completion time is smallest among peers in S'. By retrieving block j as fast as possible, the remainder $T_p(j) - T_f(j)$ can be used to retrieve the succeeding blocks from distant peers or peers with insufficient bandwidth.

SR (Select Reliable) Method selects a peer with the lowest possibility of block disappearance among those in S'. Since the capacity of a cache buffer is limited, block j may be replaced by another block before a request for block j arrives at the provider peer. The list of block identifiers in a response message is in ascending order of referenced time. Thus, a block located closer to the head of the list is likely to be removed in the near future. In SR method, in order to perform reliable retrieval, we consider the peer with a buffer in which block j has the largest number among those of peers in S'.

Step 5 Derive estimated completion time of retrieval $T_f(j)$ and time $T_r(j)$ to send a request message for block j as follows.

$$T_f(j) = \max(T_f(j-1), T_{now} + R(P(j))) + \frac{B(j)}{A(P(j))} \tag{3.2}$$

$$T_r(j) = T_f(j) - R(P(j)) - \frac{B(j)}{A(P(j))} \tag{3.3}$$

Step 6 If $j = kN$, finish and wait for receiving the next response message. Here, k is the round number. Otherwise, set $j \leftarrow j + 1$ and go back to Step 2.

A peer emits a request message for block j to peer $P(j)$ at $T_r(j)$ and sets r to j. On receiving the request, peer $P(j)$ initiates block transmission. If it replaced block j with another block since it returned a response message, it informs the peer of a cache miss. When a cache miss occurs, the peer determines another provider peer based on the above algorithm. However, if it has already requested any block after

j, it gives up retrieving block j in order to keep the media play-out in order.

After receiving block j, the peer replaces $T_f(j)$ with the actual completion time. In the algorithm, the estimated completion time of retrieval of block j depends on that of block $j-1$, as in Eq. (3.2). Therefore, if the actual completion time $T_f(j)$ of the retrieval of block j changes because of changes of network conditions or estimation errors, the peer applies the algorithm and determines provider peers for succeeding blocks. Our proposed algorithm stated above depends on the accuracy of estimation. One of possible solutions to inaccurate estimates is to introduce some reserved time in Eq. (3.2). In addition, deferment of the play-out also contributes to absorption of estimation errors.

3.3 Simulation Experiments

We conducted simulation experiments to evaluate the basic characteristics of our proposed methods in terms of the amount of search traffic and the continuity of media play-out.

We used a P2P logical network with 100 peers randomly generated by the Waxman algorithm [65] with parameters $\alpha = 0.15$ and $\beta = 0.3$. An example of generated networks is shown in Fig. 3.3. The round trip time between two contiguous peers is also determined by the Waxman algorithm and ranges from 10 ms to 660 ms. To investigate the ideal characteristics of our proposed methods, the available bandwidth between two arbitrary peers does not change during a simulation experiment and is given at random between 500 kbps and 600 kbps, which exceeds the media coding rate of CBR 500 kbps.

At first, none of the 100 peers watch any media stream. Then, peers randomly begin to request a media stream one by one. The inter-arrival time between two

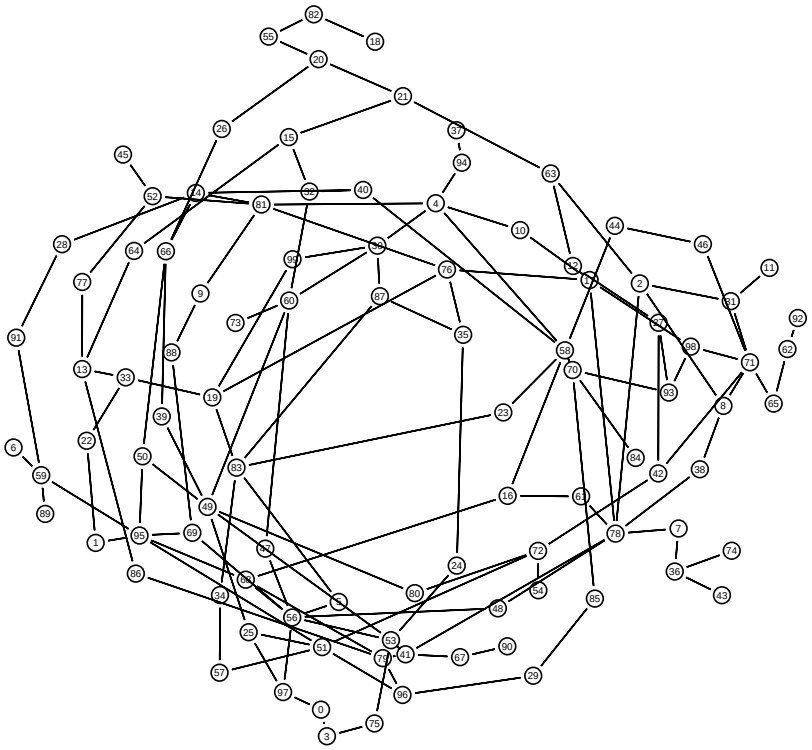


Figure 3.3: Random network with 100 peers

successive media requests for the first media stream among clients follows an exponential distribution whose average is 20 minutes. Forty media streams of 60 minutes length are available. Media streams are numbered from 1 (most popular) to 40 (least popular), where the various levels of popularity follow a Zipf-like distribution with $\alpha = 1.0$. Therefore, media stream 1 is forty times more popular than media stream 40. Each peer watches a media stream without such interactions as rewinding, pausing, or fast-forwarding. When a peer finishes watching a media stream, it becomes idle during the waiting time, which also follows a exponential distribution whose average is 20 minutes. A media stream is divided into blocks of 10-sec duration and 625 KBytes. Each peer sends a query message for a succession of six blocks, i.e., $N = 6$, and retrieves blocks. Blocks obtained are deposited into a cache buffer of

675 MB, which corresponds to three media streams. At the beginning of each simulation experiment, each peer stores three whole media streams in its cache buffer. The initial population of each media stream in the network also follows a Zipf-like distribution whose parameter α is 1.0. To prevent the initial condition of the cache buffer from influencing system performance, we only use the results after the initially cached blocks are completely replaced with newly retrieved blocks for all peers.

We consider six combinations of three search methods, i.e., full flooding only, FL, and FLS, and two retrieval methods, i.e., SF and SR. We conducted 90 set of simulations for each of six methods and show average values in the following figures.

3.3.1 Evaluation of Scalability of Search Methods

First, we evaluate search methods from the viewpoint of the scalability in terms of the number of queries. Figure 3.4 illustrates transitions of the average number of queries that a peer receives. As shown in Fig. 3.4, the FL method only slightly reduces the number of queries compared with full flooding. This is because the average number of relays in limited flooding is relatively large in our simulation experiments, independent of the block retrieval method. Since TTL is determined in accordance with the previous search results, the number of relays chosen for limited flooding immediately after full flooding tends to remain large. On the other hand, selective search can considerably reduce the number of queries.

3.3.2 Evaluation of Continuous Media Play-out

We defined the waiting time as the time between the emission of the first query message for the media stream and the beginning of reception of the first block. Although not shown in figures, we observed that, independent of the combination of methods,

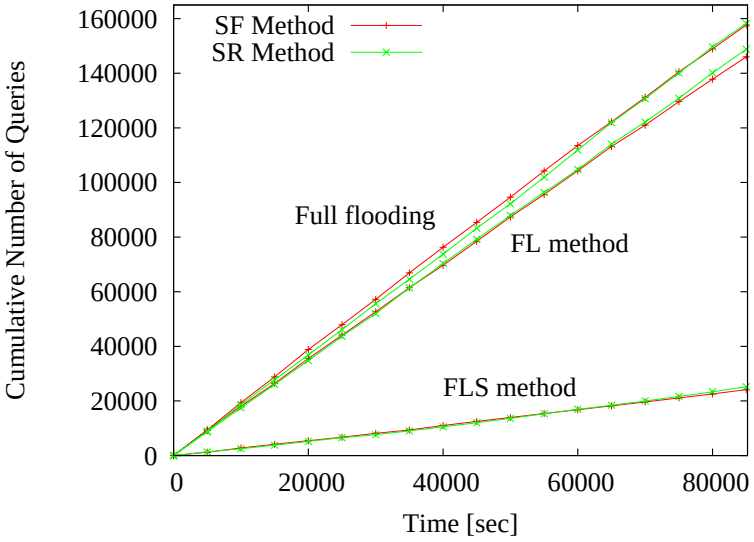
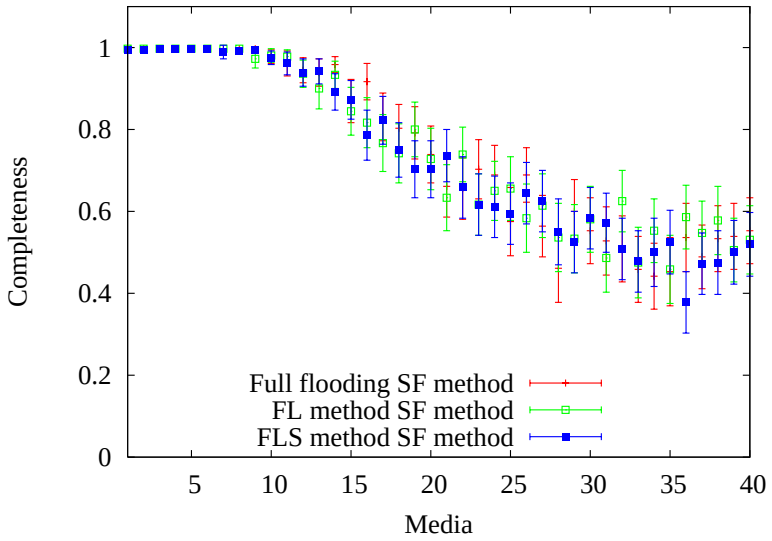


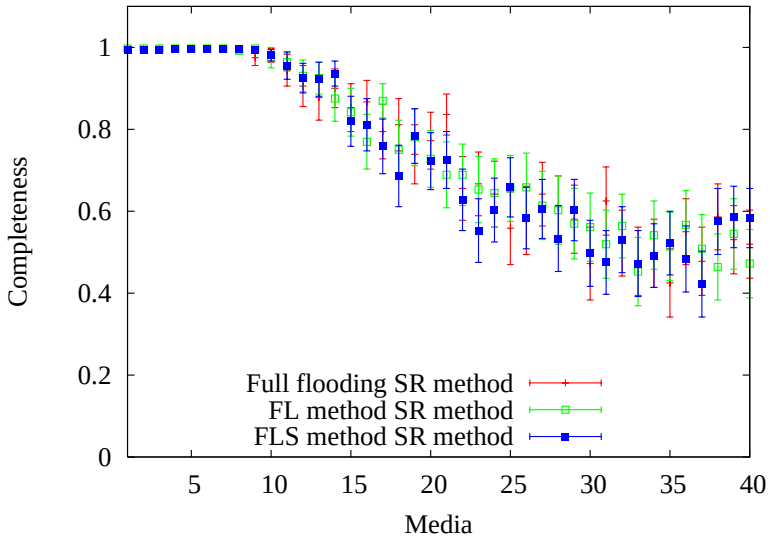
Figure 3.4: Cumulative number of queries

the waiting time decreases as the popularity increases, and, independent of popularity, all media streams that are successfully found can be played out within 3.5 sec. This is small enough from a viewpoint of service accessibility [66].

Figures 3.5(a) and 3.5(b) illustrate the completeness with 95 % confidence interval of each media stream after 20000 media requests occur. To evaluate the continuity of media play-out, we define the completeness as the ratio of the number of retrieved blocks in time to the number of blocks in a media stream. As shown in Figs. 3.5(a) and 3.5(b), independent of method, media streams from 1 to 10 are played out almost continuously from the beginning to the end. On the other hand, as media popularity decreases, the completeness also deteriorates. In our experiments, most of the blocks that cannot be retrieved in time are blocks that have already been replaced by blocks of more popular streams. In spite of the less number of query messages, FLS method can accomplish equivalent completeness compared with other two methods. Comparing Figs. 3.5(a) and 3.5(b), we find that there is little difference between SF and SR. This is because the remaining time is not used effectively and unexpected cache miss hardly occurs in our experiments.



(a) SF method



(b) SR method

Figure 3.5: Completeness

3.4 Conclusion

We first propose scalable block search methods taking into account temporal order of reference to media blocks that means users usually play out a media stream from

the beginning to the end. Then, we also propose block retrieval methods to accomplish continuous media play-out. Through several simulation experiments, we show that the FLS method can provide users with continuous media play-out for popular media streams while reducing the search traffic to $\frac{1}{6}$ compared with the full flooding. Simulation results also show that the cache replacement algorithm is important to improve the completeness of unpopular media streams.

Chapter 4

Adaptive and Robust Media Streaming on P2P Networks

4.1 Introduction

In the previous chapter, we have shown that our systems can accomplish continuous media play-out for popular media streams without introducing extra load on the system through several simulation experiments. However, we have also found that the completeness of media play-out deteriorates as the media popularity decreases. The reason is that popular media streams are cached excessively while unpopular media streams eventually disappear from the network. Although Least Recently Used (LRU) is a simple and widely used cache replacement algorithm, it fails in continuous media play-out.

To improve the completeness of media play-out, in this chapter we consider an effective cache replacement algorithm that takes into account the supply and demand for media streams. Since there is no server, a peer has to make conjectures about the behavior of other peers by itself. A peer estimates the supply and demand from

P2P messages that it relays and receives from a flooding-based media search. Then a peer determines a media stream to discard to make room for a newly retrieved block. Furthermore, a peer also adapts to changes in the supply and demand of media streams. For this purpose, we propose a novel caching algorithm based on the response threshold model of division of labor and task allocation in social insects [52].

In biology, social insects, such as ants, also construct a distributed system [53]. In spite of the simplicity of their individuals, the insect society presents a highly structured organization. It has been pointed out that social insects provide us with a powerful metaphor for creating decentralized systems of simple interacting [53, 54]. In particular, a recently proposed model of division of labor in a colony of primitively eusocial wasps, based on a simple reinforcement of response thresholds, can be transformed into a decentralized adaptive algorithm of task allocation [52]. By regarding the replacement of media streams as a task, we propose a fully distributed and autonomous cache replacement algorithm which can adapt to changes in environments, i.e., the supply-to-demand. Our proposed algorithm is also insensitive to parameter settings since it adaptively changes the response threshold taking into account the obtained information from the network. Through several simulation experiments, we evaluate the algorithm in terms of the completeness of media play-out, adaptability to changes in media popularity, and sensitivity to parameter settings.

Furthermore, in an actual situation, media streaming fails since peers participating a service occasionally leave a P2P network due to user's interactions or system failures. Network conditions including the available bandwidth and Round Trip Time (RTT) also change dynamically. In this chapter, we newly propose a block retrieval method which dynamically switches provider peers based on the estimation of the available bandwidth and RTT. In addition, we extend our block search method to prepare for peer leaves. Through several simulation experiments, we evaluate our proposed methods in terms of the completeness of media play-out under unstable

system conditions.

The rest of this chapter is organized as follows. In Section 4.2, we propose a supply-demand-based cache replacement algorithm, an adaptive and robust block search method, an adaptive block retrieval method. Next, we evaluate our proposed methods through several simulation experiments in Section 4.3. Finally, we conclude the results of our study in Section 4.4.

4.2 Adaptive and Robust P2P Media Streaming

4.2.1 Supply-Demand-based Cache Replacement Algorithm

Although LRU is a simple and widely used scheme, simulation results showed that LRU cannot accomplish continuous media play-out under the condition of heterogeneous media popularity. This is because popular media streams are cached excessively while unpopular media streams eventually disappear from the P2P network.

In this section, to solve this problem, we propose a bio-inspired cache replacement algorithm that considers the balance between supply and demand for media streams. Since there is no server in an unstructured P2P network, a peer has to make conjectures about the behavior of other peers by itself. It is important to avoid the situation where a peer aggressively collecting information on supply and demand by communicating with other peers, since this brings extra load on the system and deteriorates the system scalability. Therefore, in our scheme, a peer estimates them based on locally available and passively obtained information, i.e., search results it obtained and P2P messages it relayed. Then, each peer autonomously determines a media stream to replace so that the supply and demand is well-balanced according to the media popularity in the network. For this purpose, we use the response threshold model [52].

In the response threshold model of the division of labor, the ratio of individuals that perform a task is adjusted in a fully-distributed and self-organizing manner. The demand to perform a task increases as time passes and decreases when it is performed. The probability that individual i performs a task is given by the demand, i.e., stimulus s, and response threshold θ_i as $\frac{s^2}{s^2+\theta_i^2}$, for example. When individual i performs the task, θ_i is decreased and thus it tends to devote itself to the task. Otherwise, θ_i is increased. After performing the task several times, it becomes a specialist in the task. Through threshold adaptation without direct interactions among individuals, the ratio of individuals that perform a specific task is eventually adjusted to some appropriate level. As a result, they form two distinct groups that show different behaviors toward the task, i.e., one performing the task and the other hesitating to perform the task. When individuals performing the task are withdrawn, the associated demand increases. Eventually, the stimulus reaches the response thresholds of the individuals in the other group, i.e., those not specialized for that task. Some individuals are stimulated to perform the task, their thresholds decrease, and finally they become specialized for the task. Finally, the ratio of individuals allocated to the task reaches the appropriate level again.

4.2.2 Description of Algorithm

By regarding the replacement of media streams as a task, we propose a cache replacement algorithm based on the response threshold model. In the cache replacement, a task corresponds to discarding a block of a media stream. However, per-block based decision consumes much computational power and memory. In addition, it leads to fragmentation of cached streams, and a cache becomes a miscellany of variety of independent blocks of media streams. Thus, we define a stimulus as the ratio of supply to demand for a media stream. By introducing the response threshold model,

a peer continuously replaces blocks of the same stream with newly retrieved blocks once a stream is chosen as a victim, i.e., a media stream to be replaced. As a result, fragmentation of media streams can be avoided. Each peer discards blocks based on the following algorithm when there is no room in the cache buffer to store a newly retrieved block.

Step1 Estimate the supply and demand for media streams per round. For a set of cached media streams M, a peer calculates supply $S(i)$ and demand $D(i)$ for media stream $i \in M$ from search results it received and messages it relayed at the previous round. $S(i)$ is the ratio of total number of blocks for media stream i in received and relayed response messages to the number of blocks in media stream i. Here, to avoid overestimation, only response messages received are taken into account for $S(i)$ when a peer watches stream i. $D(i)$ is the number of query messages for media stream i, which the peer emitted and relayed.

Step2 Determine a media stream to replace. Based on the "division of labor and task allocation", we define ratio $P_r(i)$ that media stream i is replaced as follows:

$$P_r(i) = \frac{s^2(i)}{s^2(i) + \theta^2(i) + l^2(i)}, \tag{4.1}$$

where $s(i)$ is derived as $\max\left(\frac{S(i)-1}{D(i)}, 0\right)$, which indicates the ratio of supply to demand for media stream i after the replacement. $s(i)$ means how excessively media stream i exists in the network after it is discarded. $l(i)$ is the ratio of the number of locally cached blocks to the number of blocks in media stream i. $l(i)$ is used to restrain the replacement of a fully or well-cached stream. Among cached streams except for the stream being watched, e.g., stream m, a victim is chosen with probability $\frac{P_r(i)}{\sum_{i \in M-m} P_r(i)}$. Then, a peer discards blocks from the head or the tail of the stream at random. As in [54], thresholds are regulated

using Eq.(4.2). Thus, media i is to be discarded more often once it is chosen as a victim.

$$\forall j \in M,\ \theta(j) = \begin{cases} \theta(j) - \xi & \text{if } j = i \\ \theta(j) + \varphi & \text{if } j \neq i \end{cases} \tag{4.2}$$

Inspired by biological systems, we can accomplish fully distributed but globally well-balanced cache replacement. Furthermore, our proposed algorithm is insensitive to parameter settings since it adaptively changes the response threshold in accordance with the obtained information from the network. With slight modification of equations of the response threshold model, we can apply our proposed algorithm to other caching problems in distributed file sharing systems.

4.2.3 Adaptive Block Retrieval Method

In an actual situation, system conditions dynamically change. The in-time block retrieval fails if the available bandwidth decreases due to congestions. A peer cannot find a provider peer since search results becomes unreliable due to peer failures or leaves. To tackle this problem, we propose an adaptive block retrieval method that appropriately switches provider peers based on the estimated available bandwidth and RTT. As far as the following condition holds, a peer can retrieve block j in time.

$$\frac{V(t) + r(j,t)}{\frac{B_s}{B_t}} \geq \frac{r(j,t)}{\Delta_{bw} A(i,t)}, \tag{4.3}$$

where $A(i,t)$ is the available bandwidth from peer i at time t, $V(t)$ is the remaining buffer size at t, $r(j,t)$ is the remaining size of block j being retrieved at t. B_s and B_t are block size and block time, respectively. Δ_{bw} takes into account the degree of accuracy of bandwidth estimation. When peer i leaves a P2P network, the available

bandwidth is considered to be zero.

When Eq.(4.3) cannot be satisfied, the peer tries to find an alternative provider peer i' from which it can retrieve the remainder of block j in time. Peer i' must satisfy the following condition:

$$\frac{r(j,t)}{A(i,t)} \geq \min_{i' \in S_j - i} \left(\frac{r(j,t)}{A(i',t)} + 2R(i',t) \right), \tag{4.4}$$

where S_j is the set of provider peers of block j obtained from the search results and $R(i',t)$ is the RTT from peer i' at time t. If the peer can find i' then it retrieves the remainder of block j from i'. Otherwise, it gives up retrieving block j.

4.2.4 Adaptive and Robust Block Search Method

If any of provider peers leaves a P2P network, a peer loses a chance to find and retrieve the corresponding block. To improve the robustness, we introduce parameter x, which defines the number of provider peers to be found in the current round to move to the selective search in the next round. For example, by setting x to two, a peer moves to the selective search when it finds two provider peers and it can prepare an alternative provider peer.

4.3 Simulation Experiments

We used the same simulation model used in the previous Chapter. Based on the values used in [53], we set the parameters of the cache replacement algorithm as follows: $\xi = 0.01$ and $\varphi = 0.001$. $\theta(i)$ was initially set to 0.5, but it dynamically changed between 0.001 and 1. $s(i)$ was normalized by dividing by $\sum_i s(i)$. To prevent the initial condition of the cache buffer from influencing system performance, we only used the results after the initially cached blocks were completely replaced with newly

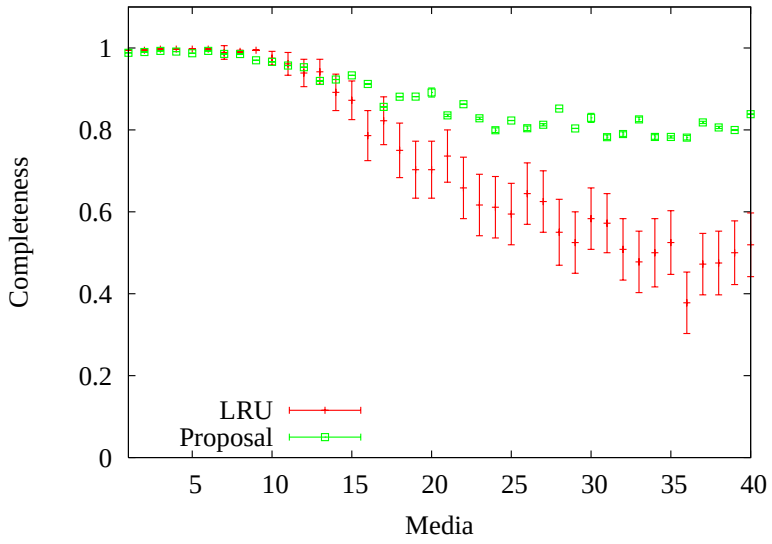


Figure 4.1: Completeness (LRU vs. Proposal)

retrieved blocks.

4.3.1 Evaluation in Stable P2P Networks

We conducted simulation experiments to evaluate our proposed cache replacement algorithm in terms of the completeness of media play-out, adaptability to changes in media popularity, and sensitivity to parameter settings. In this scenario, we considered the stable P2P network where the available bandwidth and RTT did not change and we set x to 1 and employed the in-time block retrieval method. We set the RTT from 10 ms to 660 ms and the available bandwidth from 500 kbps to 600 kbps between two arbitrary peers. We show the average values of 40 sets of simulations in the following figures.

Evaluation of Continuous Media Play-out

We define the waiting time as the time between the emission of the first query message for the media stream and the beginning of reception of the first block. Although not shown in the figures, we observed that the waiting time decreases as the popularity increased. However, independent of popularity, all media streams successfully found can be played out within 1.7 sec.

We define the completeness as the ratio of the number of retrieved blocks in time to the number of blocks in a media stream. Figure 4.1 depicts the completeness with a 95 % confidence interval of each media stream after 20000 media requests. We find that our proposed algorithm can improve the completeness of unpopular media streams without affecting popular streams. This is because the amount of cached streams for popular media streams is suppressed in accordance with their popularities. As time passes, the completeness for unpopular media streams slightly decreases even with our algorithm due to the disappearance of blocks. We expect that a media streaming server that stores the original media streams can provide users with the remaining part of unpopular media streams. Since the media popularity follows the Zipf distribution, 70% requests concentrate on the popular media streams whose completeness is one. The remaining 30% requests require to retrieve at most 20% blocks of an unpopular media stream from the server. This means that our proposed methods can accomplish effective media streaming while considerably reducing the load on the server.

Evaluation of Adaptability to Changes in Popularity

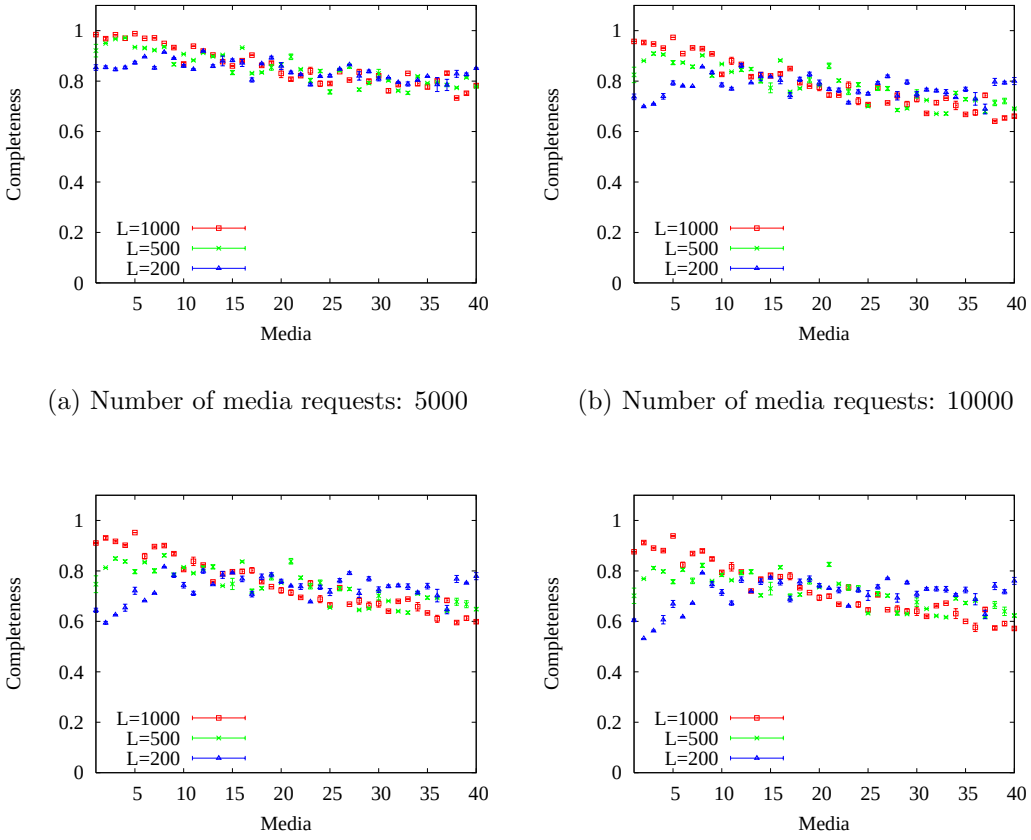
We changed the popularity of each media stream over time based on a model used in [15]. In the model, the media popularity changes every L media requests. Another well-correlated Zipf-like distribution with the same parameter ($\alpha = 1.0$) is used for

the change. The correlation between two consecutive Zipf-like distributions is modeled by using a parameter n that can be any integer between 1 and the number of media streams, i.e., 40. First, the most popular media stream in the current Zipf-like distribution becomes the r_1th popular in the next Zipf-like distribution, where r_1 is randomly chosen between 1 and n. Then, the second popular media stream in the current distribution becomes the r_2th popular in the next distribution, where r_2 is randomly chosen between 1 and $\min(40, n+1)$, except that r_1 is not allowed. Thus, as time passes, initially popular media streams become less popular while initially unpopular media streams become more popular. We set $n = 5$ in the experiments and the demand changes every L media requests.

Figure 4.2 illustrates the transition of the completeness of the proposed algorithm. To clarify the transition, we show the completeness at instants when 5000, 10000, 15000, and 20000 media requests occur. To evaluate the adaptability to the speed of popularity change, we set L to 200, 500, and 1000. As shown in Fig. 4.2, in the case of $L = 200$ where the popularity changes fast, the completeness of initially unpopular media streams, identified by a large number, becomes higher than that of initially popular media streams with a smaller number as time passes and demand changes. On the other hand, in the case of $L = 1000$, where the popularity changes rather more slowly, the completeness of media streams with a small number is kept higher than that of media streams with a large number. Thus, we can conclude that our proposed algorithm can adapt to changes in media popularity.

Evaluation of Sensitivity to Parameter Settings

Our cache replacement algorithm has a set of parameters, $\theta(i)$, ξ, and φ. $\theta(i)$ is dynamically adjusted by Eq.(4.2). We conducted several simulation experiments by changing ξ and φ in Eq. 4.2, which are associated with the degree of adherence to a specific victim in cache replacement. Although not shown in figures, we found



(a) Number of media requests: 5000

(b) Number of media requests: 10000

(c) Number of media requests: 15000

(d) Number of media requests: 20000

Figure 4.2: Completeness with changes in media popularity ($\xi = 0.01$, $\varphi = 0.001$)

that there was almost no difference among different ξ and φ. It follows that the proposed algorithm is insensitive to parameter settings. Thus, we do not need to give careful consideration to the problem of parameter setting as in other algorithms that need several critical parameters to be carefully determined in advance. Furthermore, the proposed algorithm flexibly adapts to changes in media popularity without any parameter adjustment.

4.3.2 Evaluation in Unstable P2P Networks

We conducted simulation experiments to evaluate the proposed improvements in terms of the completeness of media play-out under unstable P2P networks. We show the average values of 20 sets of simulations in the following figures.

Evaluation with Changes in Network Condition

We randomly changed RTT from 10 ms to 660 ms and the available bandwidth from 450 kbps and 550 kbps between two arbitrary peers at one-second intervals. A peer estimated the available bandwidth and RTT at one-second intervals with the degree of accuracy of 0.975. We set Δ_{bw} to 0.95 which was less than the estimation accuracy. Since peers did not leave in this scenario, we set x to 1.

Figure 4.3 depicts the completeness of the in-time block retrieval method and the adaptive block retrieval method. For comparison purposes, results in a stable environment are also shown. As shown in this figure, the completeness of the in-time block retrieval method is less than 0.5 even for the most popular media stream under unstable network conditions. This is because peers keep retrieving blocks even when the available bandwidth decreases and they cannot finish the retrieval in time. In contrast, the adaptive block retrieval method can improve the completeness by appropriately changing a provider peer. However, it is still lower than in a stable environment.

For a peer effectively to switch provider peers, it needs to have at least one alternative peer i' which satisfies Eq.(4.5).

$$\left\{ i' \in S_j - i \left| \frac{V(t) + r(j,t)}{\frac{B_s}{B_t}} \geq \frac{r(j,t)}{A(i',t)} + 2R(i',t) \right. \right\} \tag{4.5}$$

The number of alternative peers depends on the number of media streams against the

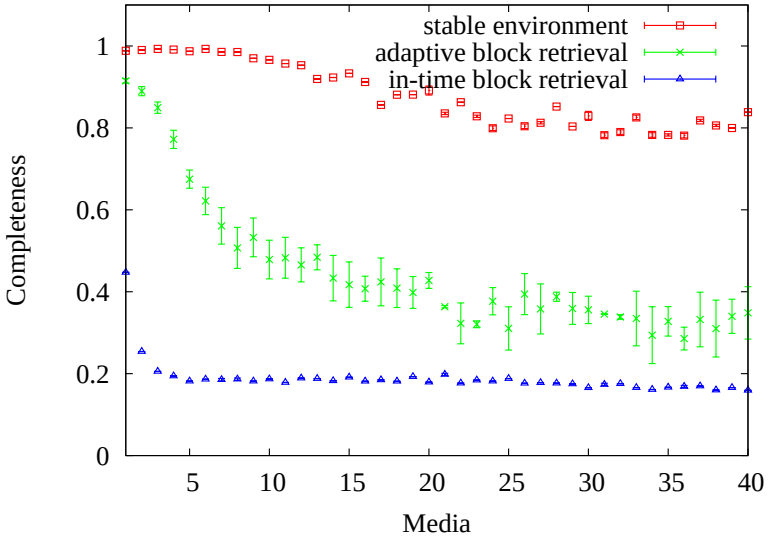


Figure 4.3: Completeness (40 streams)

capacity of the whole P2P network. Figure 4.4 illustrates the results of the case with 10 media streams. As shown in the figure, the completeness of the adaptive block retrieval method is more than 0.9 independently of the media popularity. Although not shown in figures, we conducted additional experiments with the various number of media streams and found that the completeness of more than 0.9 could be attained for all media popularity when the number of media streams was less than 12 against the network capacity of $100 \times 3 = 300$ media streams.

Evaluation with Leaves of Provider Peers

To evaluate the robustness to peer leaves, we next considered the following scenario. First, one designated peer was randomly chosen. Then, we removed peers while the designated peer was retrieving the first media stream. Peers to be removed were randomly chosen among peers that deposited blocks of the media stream which the designated peer was interested in. The inter arrival time between two successive

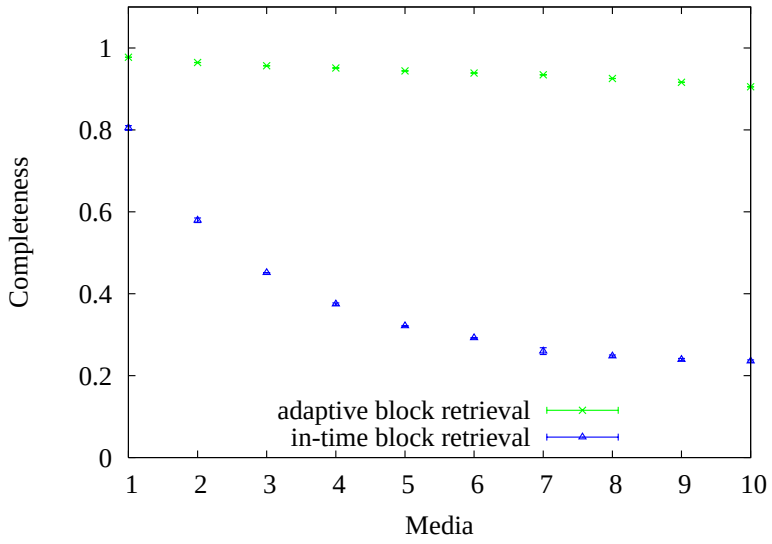


Figure 4.4: Completeness (10 streams)

removals followed an exponential distribution whose average was 10 minutes. To investigate the impact of peer leaves, there was no reconstruction or recovery of the P2P network when peers left and links were broken and the available bandwidth and RTT did not change in this scenario.

Figure 4.5 shows that the completeness of $x = 1$ is higher than that of $x = 2$, on the contrary to our expectation. With $x = 2$, a peer conducts the limited flooding more often than the case with $x = 1$. In the limited flooding, query messages are diffused over a P2P network by being relayed by peers. Thus, the possibility that a peer can find an appropriate provider peer decreases due to breaks of the network caused by disappearance of peers. On the other hand, in the selective search, query messages are directly sent to provider peers without mediations of other peers. As a result, even though only 6 % peers left the network in Fig. 4.5, the completeness of the case of $x = 1$ becomes superior to that of $x = 2$. Thus, we can conclude that the selective search is more robust to peer leaves than the limited flooding from

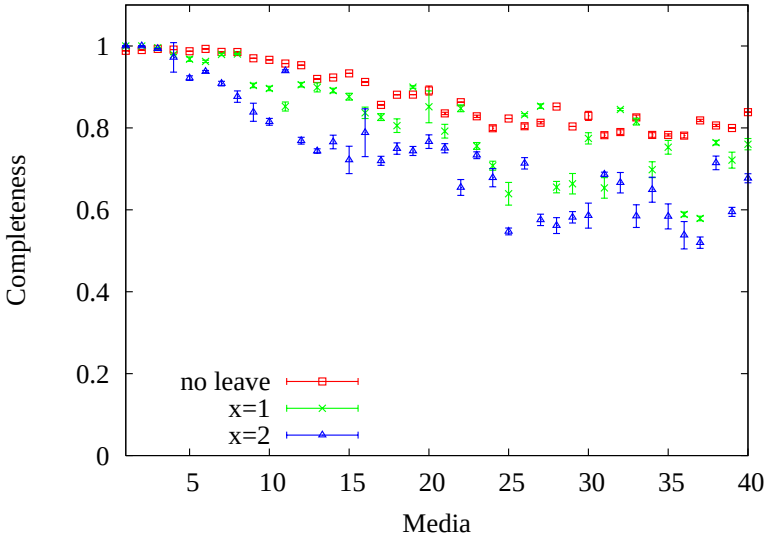


Figure 4.5: Completeness with peer leaves

simulation results.

4.4 Conclusion

In this chapter, we proposed an adaptive and robust block search method, an adaptive block retrieval method, and a supply-demand-based cache replacement algorithm inspired by biological systems. Through several simulation experiments, we showed that our proposed cache replacement algorithm could accomplish continuous media play-out independent of media popularity and adapt to changes in media popularity. Furthermore, we demonstrated that the adaptive block retrieval method could improve the completeness compared to the in-time block retrieval method. In addition, we found that the selective search was more efficient than flooding methods when peers failed and left.

Chapter 5

Construction Methods of a Low-Diameter and Location-Aware P2P Network

5.1 Introduction

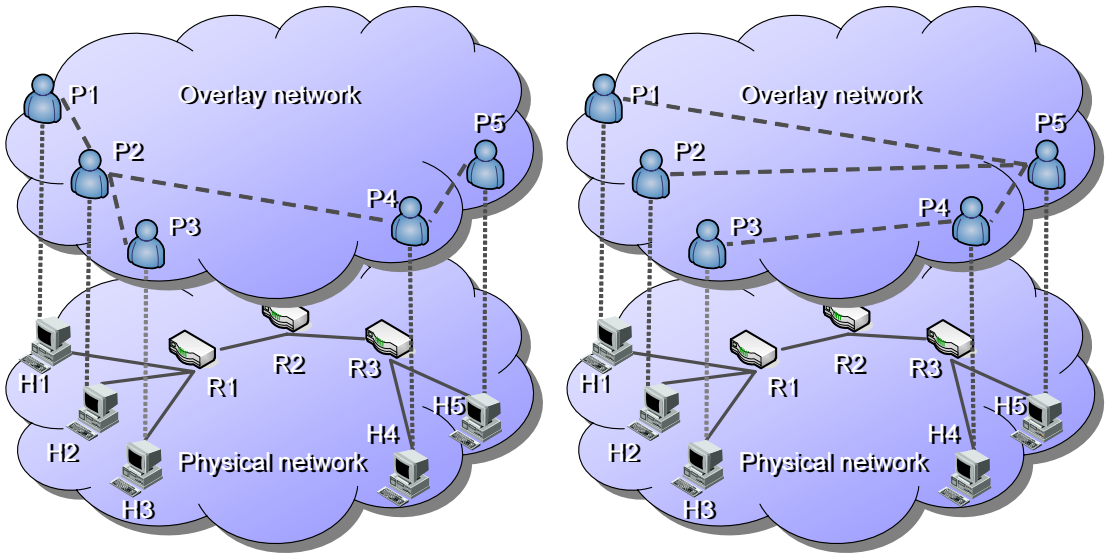
In recent years, P2P file-sharing systems have become widely diffused. There are three kinds of architectures for P2P systems: centralized, decentralized-structured, and decentralized-unstructured. Since decentralized-unstructured P2P systems, such as Gnutella [41] and KaZaA [42], are the most popular in the current Internet, in this thesis, we also focus on a decentralized-unstructured model. In decentralized-unstructured P2P systems, there is no server that manages meta information on peer and file locations. A peer tries to find its desired file by flooding a query in a P2P overlay network. From a set of peers in search results obtained by the query, it determines a provider peer from which it retrieves a file.

The structure of an overlay network determines the effectiveness of search in terms

of network load and user QoS. An inefficient overlay network induces unnecessary and huge traffic into underlying physical networks. Specifically, the diameter of an overlay network affects the number of hops, i.e., a TTL value, required to find a sufficient number of provider peers and files desired by users. Furthermore, if an overlay network is constructed without taking into account the topology of underlying physical networks, a logical link may be established between physically-distant peers. Passing a message from one peer to another takes time and much network resources. A peer has to wait for a long time to obtain information on physically-close providers from which it can quickly retrieve its desired file.

To tackle these problems, an overlay network should reflect the characteristics of the underlying physical topology, e.g., the degree distribution and the physical proximity. Figure 5.1 illustrates examples of overlay networks constructed on a physical network. The physical network consists of five hosts ($H1 \sim H5$) and three routers ($R1 \sim R3$). Peers ($P1 \sim P5$) corresponding to the hosts. The overlay network in Fig. 5.1(a) has three physically-close logical links while that in Fig. 5.1(b) has only one physically-close link. As a result, peers on Fig. 5.1(a) can find physically-closer peers with faster response time than those on Fig. 5.1(b). This contributes to reduction of the network load while improving the user QoS in terms of the search and retrieval time of the desired file.

In this thesis, we propose novel methods to construct a low-diameter and location-aware overlay network where peers can find physically-close providers without introducing much load on underlying physical networks. There have been several research works on the construction of a low-diameter network. Barabási-Albert (BA) model [56] is the first and the most popular model that explains generation of a network whose degree distribution follows the power-law. The BA model is dominated by the preferential attachment (PA). The PA defines the probability P_i that node i gets links from newly added nodes as proportional to its degree k_i, that is $P_i = \frac{k_i}{\sum_{j \in S_n} k_j}$,



(a) Match (b) No match

Figure 5.1: Correlation between overlay and underlying physical network

where S_n is the set of nodes existing in the network. As a result, it constructs a scale-free network in which there are a few hub nodes that are connected with many low-degree nodes. This characteristic contributes to reduction of the diameter of the network, where a peer can discover a sufficient number of desired files in the range of a small number of logical hops.

Phenix [67] also provides an algorithm to construct a scale-free network. Although the BA model requires a centralized control mechanism, the Phenix is based on local control information. The BA model assumes that a new node knows all nodes already participating in the network when it tries to join. However, in realistic situations, a new peer only knows some of peers notified by a bootstrapping node [68, 69]. The Phenix also assumes that a new node obtains information on peers from a bootstrapping node. Then, it collects information on neighbors of the peers. Finally, it conducts the PA based on the frequency of appearance of a peer in the information.

Although the BA and Phenix can build an overlay network, they do not consider the topology of underlying physical networks. We should shorten the physical distance, i.e., the number of physical hops, between neighboring peers to reduce the load on underlying physical networks. For this purpose, in our model a peer adopts the PA to only peers that are physically close. As a result, constructed overlay network has a correlation between the logical and the physical distances. Usually in conventional P2P file-sharing systems, on receiving a response message, a peer has to examine the physical closeness of the peer by using a ping message. On the other hand, in our overlay network, logically-close peers are also physically close. It does not require for a peer to probe the physical distance from the discovered providers by emitting additional ping-like messages.

Since a new peer only knows some of peers notified by a bootstrapping node in realistic situations, it can not necessarily find appropriate peers from the initially obtained peer list and it leads to the limitation of the effectiveness of a constructed network in join phases. To improve the structure of a constructed network, a rewiring method is viable, which enables a peer to connect to more appropriate peers after the join phase [57-59].

Location-aware topology matching (LTM) is a rewiring method to build a location-aware overlay network [57]. It disconnects low productive neighbors that are found by Gnutella 0.6 protocol. In the LTM, peer P collects delay information on peers within two hops by periodically sending probing packets. Based on the delay information, P conjectures the topology of an overlay network within two hops and searches peer S that has two ore more logical paths to P. Then, P finds the most distant connection in the overlay network. If the inefficient connection is established between P and its neighbor, P cuts off the connection. Although the LTM method can construct a location-aware overlay network, it does not contribute to reduction of the diameter of the overlay network. Furthermore, the Gnutella-based random selection of the

neighbors can not necessarily connect to physically-close peers. This not only induces unnecessary traffic into underlying physical networks but also requires multiple times of rewiring to obtain physically-close neighbors.

In this thesis, we propose a new rewiring method to improve the location-awareness of an overlay network. We assume that a peer can obtain information on peers that its neighboring peers know by periodically exchanging ping-pong messages. Based on the constructed peer list, a peer finds peers that are physically closer and with a higher-degree than the current neighbors. To investigate the effectiveness of the proposed methods, we conducted several simulation experiments using real physical topologies such as Abilene and Sprint networks that also follow the power-law distribution. We evaluated diameter, physical distance to neighbors, and degree distribution of overlay networks constructed by existing and our proposed methods. Simulation results show that the proposed methods enable a peer to find physically-close providers without introducing much load into underlying physical networks.

The rest of this chapter is organized as follows. We propose construction methods of a low-diameter and location-aware overlay network in Section 5.2. Next, in Section 5.3 we evaluate our proposed methods through several simulation experiments. Finally, we conclude the thesis in Section 5.4.

5.2 Construction Methods of a Low-Diameter and Location-Aware P2P Network

In this section, we consider methods to construct an overlay network that satisfies both low-diameter and location-aware features. We first propose a BA-based construction method with a modification in the node selection for the PA in order to consider underlying physical networks. Then, we further propose a rewiring method

to improve the structure of an overlay network.

5.2.1 Tree Construction with Consideration of Underlying Physical Networks

To build a low-diameter and location-aware overlay network, a newly added peer has to connect high-degree and physically-close peers. For this purpose, we limit a set of nodes in the PA. Our proposed method is inspired by the modified BA model [69]. In the model, they introduce 'affinity' to restrict target nodes for the PA. A newly added node with a random affinity first selects nodes that are already present in the network and have a similar affinity. Then, nodes to connect are chosen among them according to the PA. Through several simulation experiments, they showed that the power-law feature was not lost by introducing the restriction. Our proposed method considers the physical distance instead of the affinity. In this thesis, we regard the number of physical hops of the shortest path as the physical distance.

When new peer i joins to an overlay network, it chooses m peers to connect among peers it knows according to the following algorithm.

1. Obtain set S_p of x peers from a bootstrapping server.

2. Calculate the physical distance to each peer in S_p by using the existing measurement tools, such as traceroute.

3. Obtain set S_c of μx peers in S_p in an ascending order of the physical distance. μ is a control parameter that ranges (0,1]. If μ is set to one, this method is equivalent to the original BA with the limitation of the size of candidate nodes. On the other hand, the smaller value of μ is intended to shorten the physical distance to neighbors in an overlay network.

4. According to the PA, select m peers in S_c. The probability $P_n(d_j)$ that peer j with degree d_j is chosen is given by:

$$P_n(d_j) = \frac{d_j}{\sum_{k \in S_c} d_k}. \quad (5.1)$$

5.2.2 Rewiring Method

In realistic situations, a newly added peer knows only some of peers from a bootstrapping node. This means that a peer can not necessarily find a peer physically close enough and with a sufficient degree at the join phase. Therefore, there is a possibility to improve the efficiency of a constructed overlay network by dynamically changing its structure. For this purpose, we propose a rewiring method where a peer first disconnects inefficient connections and then establishes connections to physically-closer and higher-degree peers. This approach is inspired by the BA model with a rewiring method described in Ref. [59]. Our rewiring method differs from the model in the selection of peers to disconnect. While the model randomly selects peers to disconnect, our rewiring method chooses the most physically-distant neighbors to disconnect.

We assume that each peer can obtain information on the other peers that are not current neighbors by exchanging ping-pong messages among neighbors in the same manner as Gnutella. In the case of Gnutella, the interval of sending ping is a few minutes. A pong message sent by a neighbor includes information on peers that the neighbor knows. Gnutella 0.6 also proposes to cache pong messages to reduce the messaging overhead. When a peer finds a new peer, it conducts the rewiring method. At first, by using a similar way to the construction method, it examines the physical distance to the new peer. Then, it chooses peers to disconnect and connect among the most distant neighboring peers and closer non-neighboring peers based on the PA. Detailed algorithm is described as follows.

1. Calculate set S_w of peers that are the most physically distant among the current neighbors. Note that neighbors whose degree are one are not included in S_w to prevent the fragmentation of an overlay network. We denote the physical distance of the peers in S_w as h_w.

2. Calculate set S_m of peers whose physical distance does not exceed h_w among non-neighboring peers.

3. Select a peer with probability $P_r(d_j)$ among peers in $S_w \cup S_m$ according to the PA as follows.

$$P_r(d_j) = \frac{d_j}{\sum_{k \in S_w \cup S_m} d_k} \tag{5.2}$$

   If the selected peer is not the current neighbor, namely a member of S_m, it is replaced with a peer randomly chosen from peers in S_w.

5.3 Simulation Experiments

To investigate the effectiveness of the proposed methods, we conducted several simulation experiments by using real physical topologies such as Abilene and Sprint networks that also follow the power-law distribution.

We evaluate our proposed methods from a view point of the structure of a constructed overlay network. We use the logical reachability and degree distribution to evaluate the diameter of an overlay network. The neighbor distance, physical cost, and provider proximity are used to evaluate how much an overlay network considers underlying physical networks. The logical reachability corresponds to the cumulative distribution function (CDF) of the number of peers existing at each logical hops. The neighbor distance is defined as the average number of physical hops between neighbors, i.e., a pair of hosts that has a logical connection. The physical cost means the

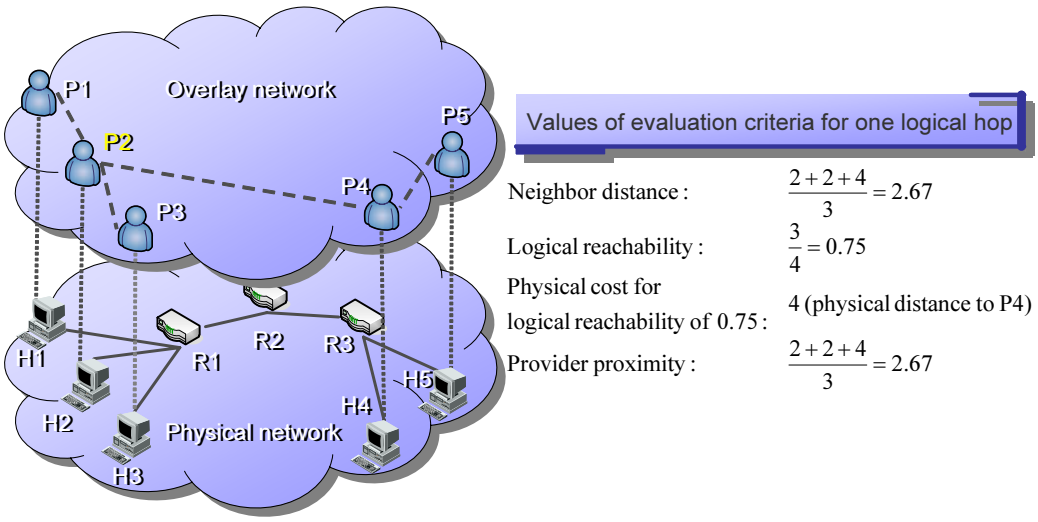
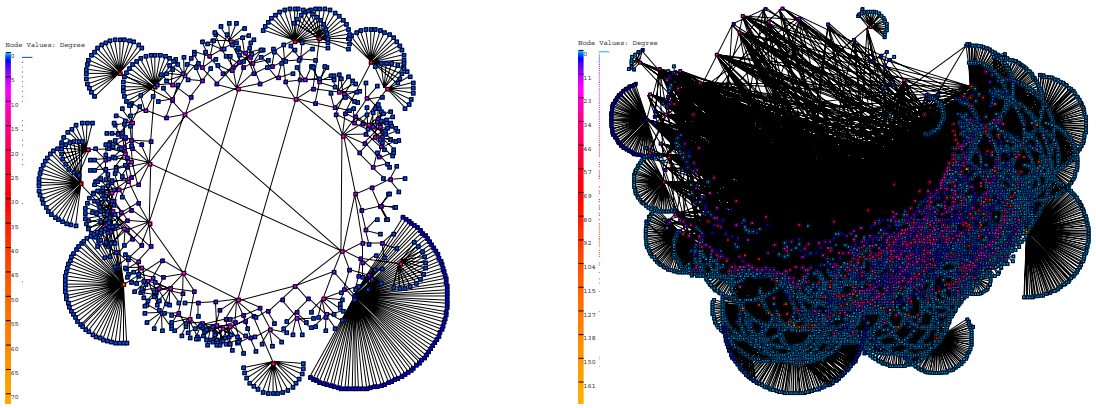


Figure 5.2: Example values of evaluation criteria

number of physical hops needed to accomplish the logical reachability. We can derive the physical cost by weighting each logical link with its physical distance, i.e., the number of physical hops between its ends. We define the provider proximity as the average physical distance from a searching peer to providers that were found within the physical cost. We used the average value among all peers for those evaluation criteria in the following simulation results. Figure 5.2 shows example values of the evaluation criteria for peer $P2$.

5.3.1 Simulation Model

We used topological data of the real physical networks: Abilene [70] and Sprint networks [71]. The Abilene network is the Internet backbone network for higher education and a part of the Internet2. It has a power-law degree distribution and forms a hierarchical structure. It is comprised of sparsely meshed core routers and many edge routers each of which is highly-connected to end users. This structure considers the router technology constraints that mean a router can have a few high

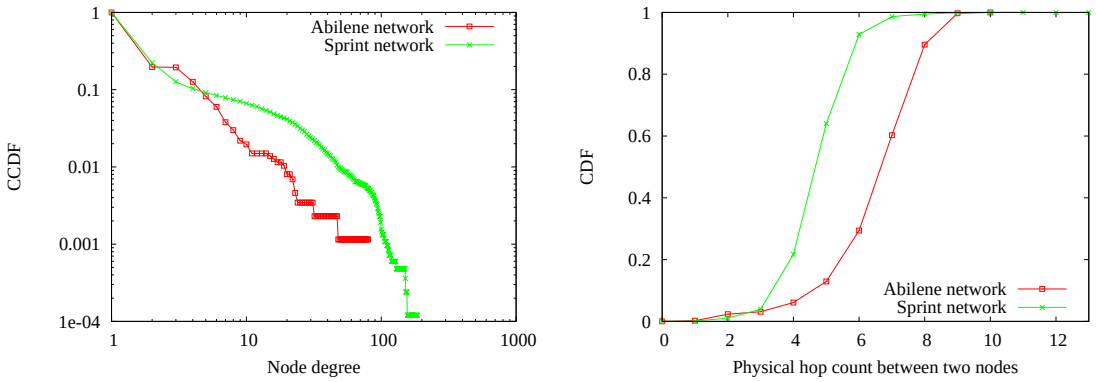


(a) Abilene network (b) Sprint network

Figure 5.3: Graphics of physical topologies

bandwidth connections or many low bandwidth connections. The Sprint network is a major ISP topology in the USA. The router level topology of the Sprint network was obtained by using a measurement tool called Rocketfuel [71]. Figures 5.3 and 5.4 illustrate physical topologies used in simulation experiments and their physical characteristics, respectively. In both physical topologies, we set peers only on the nodes whose degree is one, because we assumed that peers were end users. The number of peers in the Abilene and Sprint networks were 698 and 6478, respectively. We assumed that the latency of each physical link is identical so that we could compare our proposed methods with the delay-based LTM method.

We constructed P2P overlay networks on these physical networks by using three kinds of methods: the LTM method, BA model, and proposed methods. Since the LTM method is based on a Gnutella method, a peer periodically tries to establish connections until the number of neighbors reaches a degree limit configured in advance. We set the degree limit of each peer at 8 [57]. On the other hand, the BA model and proposed methods have no limitation on the degree. Instead, they restricted the



(a) Degree distribution

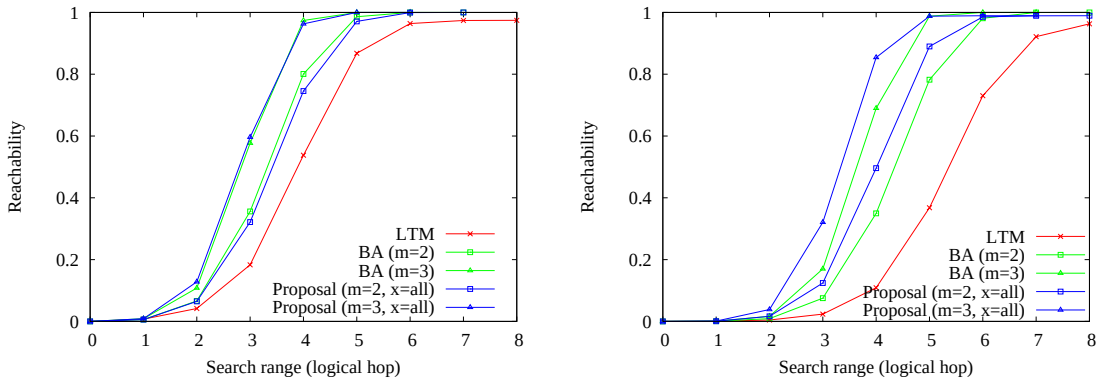
(b) Number of physical hops between two arbitrary nodes

Figure 5.4: Characteristics of physical topologies

number of connections established at the join phase to m. In the following simulation experiments, we used two values of m, that is $\lceil d_l \rceil$ and $\lfloor d_l \rfloor$. Here, d_l corresponds to the average degree of the overlay network constructed by the LTM method. In the following simulation experiments, d_l of the Abilene and Sprint networks was 5 and 4.8, respectively. We set the initial number of nodes m_0 to the same value of m. The inter-arrival time between two successive peer participations followed an exponential distribution whose average was 120 seconds. In the case of the LTM, the interval of rewiring was also set to 120 seconds. Although we conducted several simulation experiments by changing μ, i.e., the ratio of peers targeted for the PA, we only show the results of $\mu = 0.2$ in the following figures.

5.3.2 Evaluation of Basic Characteristics

We first conducted simulation experiments in the case of ideal situations where a peer could obtain information on all peers in an overlay network from a bootstrapping node. In this scenario, we evaluated the proposed method without the rewiring



(a) Abilene network

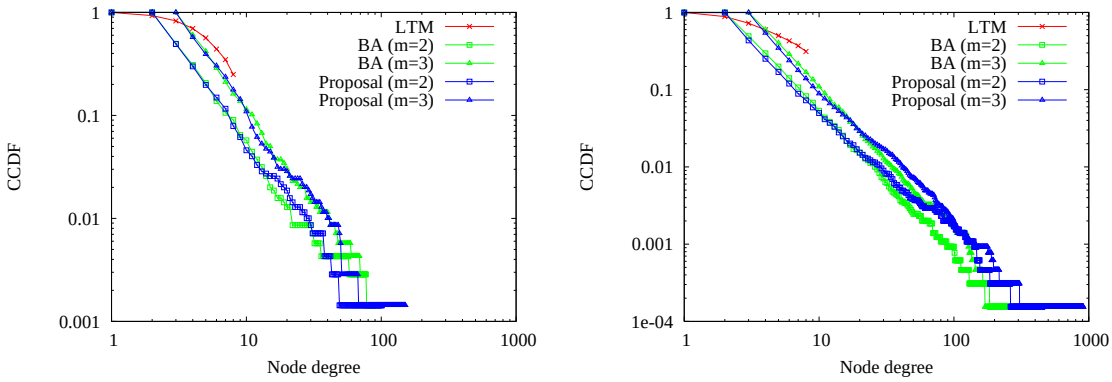
(b) Sprint network

Figure 5.5: Logical reachability ($\mu = 0.2$)

method.

First, we discuss the diameter of an overlay network. We show the logical reachability in Fig. 5.5. The proposed method and BA model can construct lower-diameter overlay networks compared with the LTM method because of their power-law features as shown in Fig. 5.6. We also find that their performance difference is larger in the Sprint network than in the Abilene network. Since there is no hub node in the overlay network constructed by the LTM method, the diameter of the overlay network is much influenced by the number of nodes. In addition, the results show that the reachability of the LTM method does not reach one due to the fragmentation of the overlay network.

Next, we focus on how much an overlay network considers underlying physical networks. Figure 5.7 illustrates the neighbor distance. The LTM method can shorten the neighbor distance compared with the BA model by disconnecting physically-distant neighbors. Our proposed method can also slightly improve the structure of an overlay network. Since the LTM can reconnect logical connections after the join



(a) Abilene network

(b) Sprint network

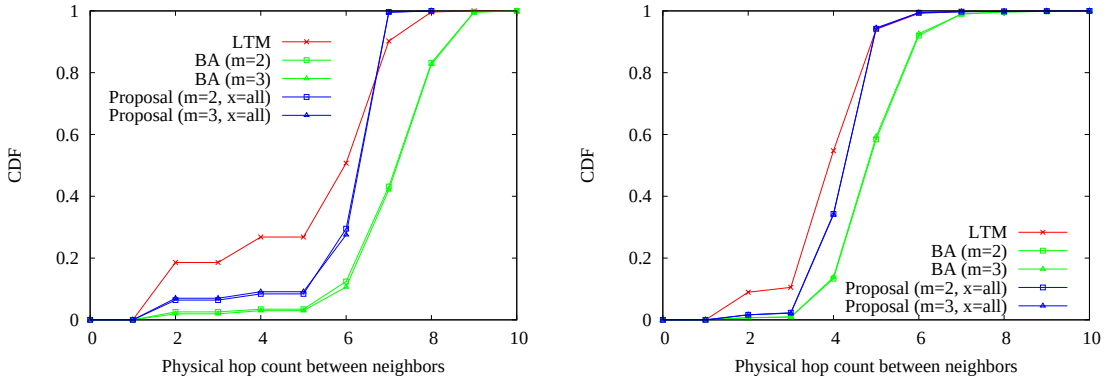
Figure 5.6: Degree distribution ($\mu = 0.2$)

phase, the LTM is superior to our proposed method without the rewiring method. Their performance difference is larger in the Sprint network than in the Abilene network. A peer can reach many peers at two physical hops in the Abilene network because edge routers connected by end users function as their hub nodes.

To consider the physical load and the effectiveness of search, we also present the physical cost in Fig. 5.8. Our proposed method is the most efficient compared with the other methods. For instance, in the case of the Sprint network, the LTM, BA ($m = 3$), and proposal ($m = 3$) need 25, 22, and 17 of physical costs to accomplish the logical reachability of 0.6, respectively.

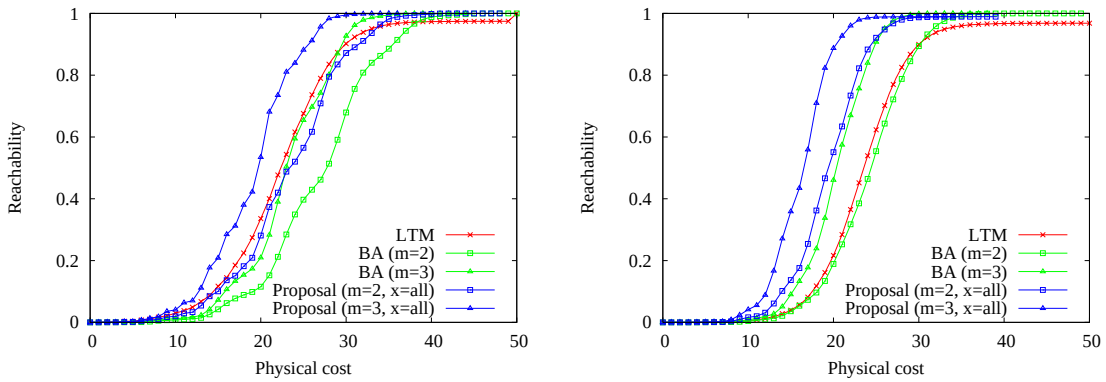
Finally, we show the provider proximity in Fig. 5.9. The proposed method and LTM method can discover physically-closer provider peers than the BA model. However, the LTM method needs more physical cost than the proposed method to find the same number of provider peers as shown in Fig. 5.8.

From these results, we found that especially in the case of the Sprint network, the proposed method could construct a low-diameter overlay network comparable with



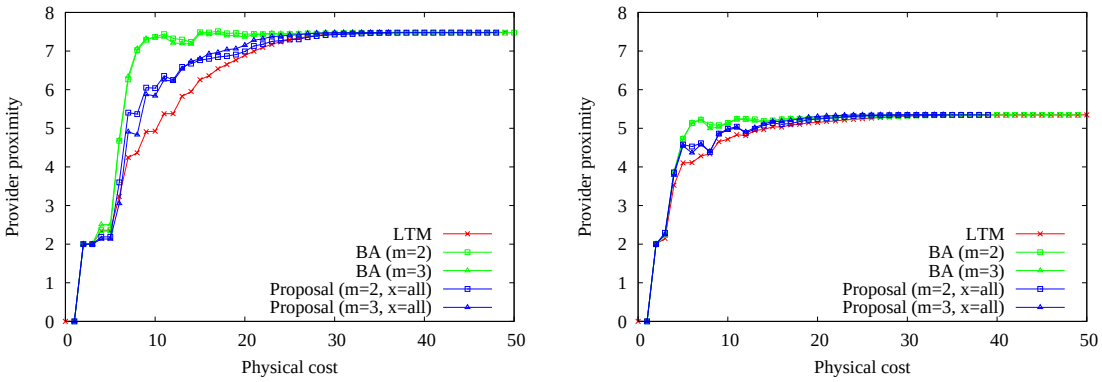
(a) Abilene network (b) Sprint network

Figure 5.7: Neighbor distance ($\mu = 0.2$)



(a) Abilene network (b) Sprint network

Figure 5.8: Physical cost ($\mu = 0.2$)



(a) Abilene network

(b) Sprint network

Figure 5.9: Provider proximity ($\mu = 0.2$)

the BA. On the other hand, there was room for improvement in the neighbor distance and physical proximity compared with the LTM method.

5.3.3 Evaluation of Rewiring Method

Next, we conducted simulation experiments in more realistic and severe environments where a new peer knows only 20 peers among 698 or 6478 peers at the join phase. The interval that a peer sent ping messages was set at 120 seconds, which was equal to the average inter-arrival time between two successive peer participations. Since it was also the same as the interval of rewiring in the LTM, there was no difference of the messaging overhead between the proposed method and LTM.

Figures 5.10 through 5.14 illustrate the results of the LTM, BA, and proposed method with and without the rewiring method. We find that the proposed method with the rewiring method is superior to the BA and LTM in terms of the logical reachability and physical cost. In addition, the neighbor distance and provider proximity are also comparable with the LTM.

Table 5.1: Correlation coefficient

	LTM	BA ($m = 3$)	Proposal with rewiring ($m = 3$)
Abilene	0.19	-0.01	0.45
Sprint	0.46	-0.17	0.42

Without rewiring, the diameter of an overlay network becomes large when x is restricted to 20 (Fig. 5.10). This is because a peer could not discover a sufficient number of physically-close hub peers in the initially obtained peer list. The rewiring method contributes to reduction of the diameter of an overlay network because the degree of hub peers increases by rewiring as shown in Fig. 5.11.

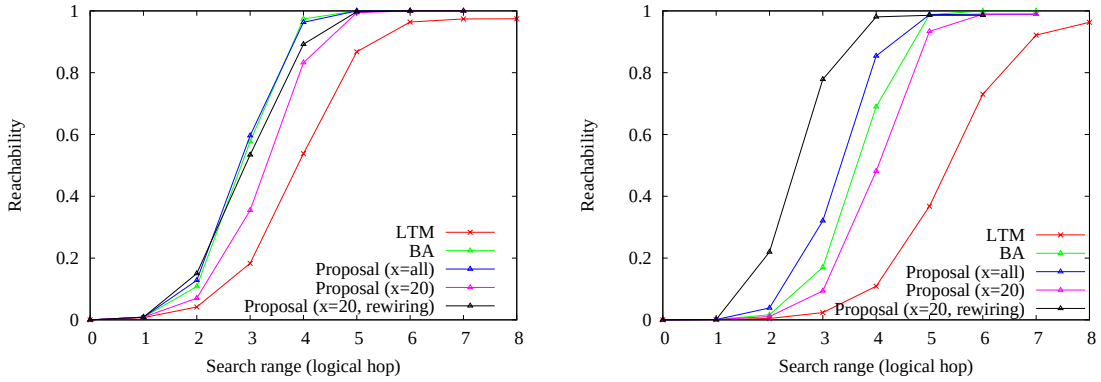
Next, as shown in Fig. 5.12, we find that the limitation of x does not affect the neighbor distance very much. Furthermore, the rewiring method contributes to reduction of the neighbor distance. By shortening the neighbor distance and the diameter of an overlay network, a peer can reduce the physical cost (Fig. 5.13) and obtain physically-close provider peers (Fig. 5.14).

Figures 5.15 and 5.16 depict the correlation between logical and physical distance. We also show the correlation coefficient in Tab. 5.1. These results also show that the proposed method can improve the correlation between the overlay and underlying physical network, that is, logically-close peers are physically close. Although the LTM method also accomplishes the almost same correlation coefficient as the proposed method in the Sprint network, it cannot construct a low-diameter overlay network. We also find that an overlay network constructed by the BA model has no correlation between the logical and physical distance.

Thus, the simulation results showed that we could construct a low-diameter and location-aware overlay network by applying the rewiring method under realistic environments.

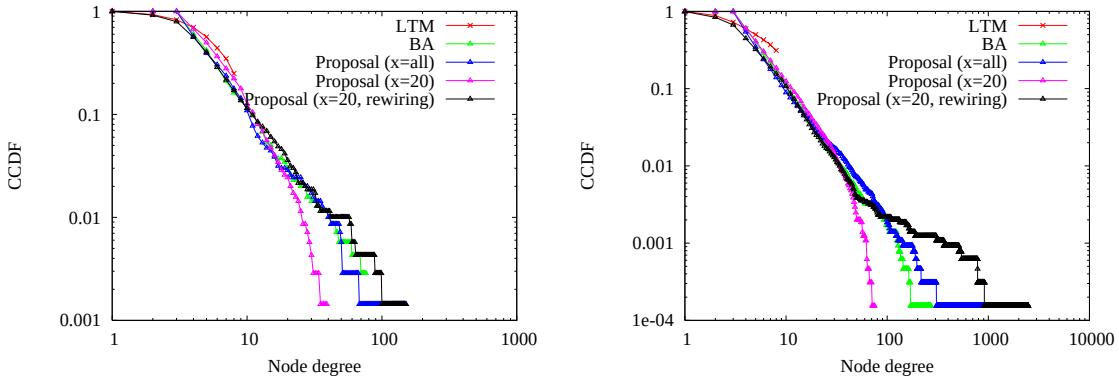
5.4 Conclusion

In this thesis, we proposed the construction method of a low-diameter and location-aware overlay network where a peer can find many physically-close provider peers. We further introduced the rewiring method to improve the structure of the constructed overlay network. Through evaluation in terms of the diameter, neighbor distance, and degree distribution of the overlay networks constructed by the existing and proposed methods, it was shown that our proposed method without rewiring could construct a low-diameter overlay network comparable with the BA model. In addition, we found that the rewiring method contributed to reduction of both the neighbor distance and diameter of the overlay network.



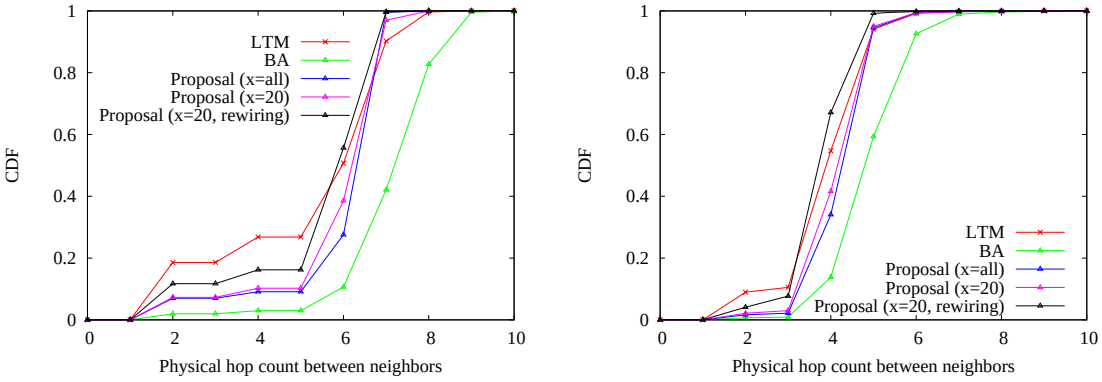
(a) Abilene network (b) Sprint network

Figure 5.10: Logical reachability ($\mu = 0.2, m = 3$)



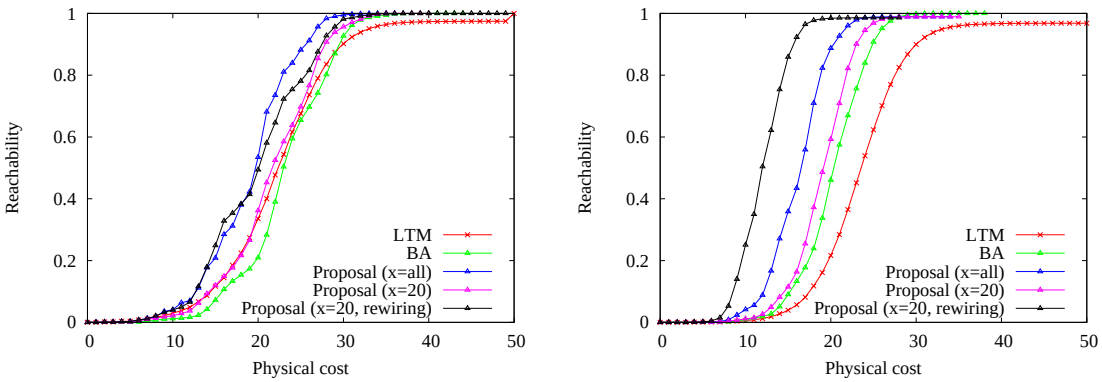
(a) Abilene network (b) Sprint network

Figure 5.11: Degree distribution ($\mu = 0.2, m = 3$)



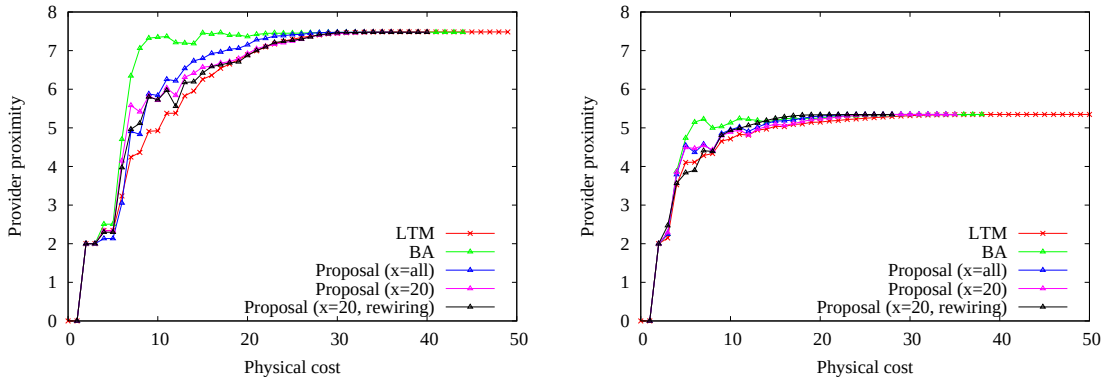
(a) Abilene network (b) Sprint network

Figure 5.12: Neighbor distance ($\mu = 0.2, m = 3$)



(a) Abilene network (b) Sprint network

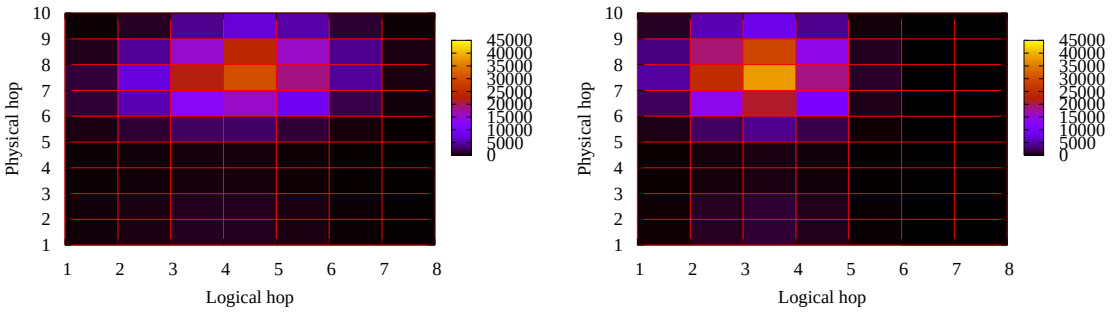
Figure 5.13: Physical cost ($\mu = 0.2, m = 3$)



(a) Abilene network

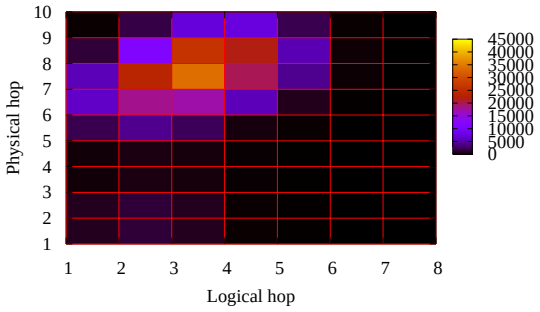
(b) Sprint network

Figure 5.14: Provider proximity ($\mu = 0.2, m = 3$)



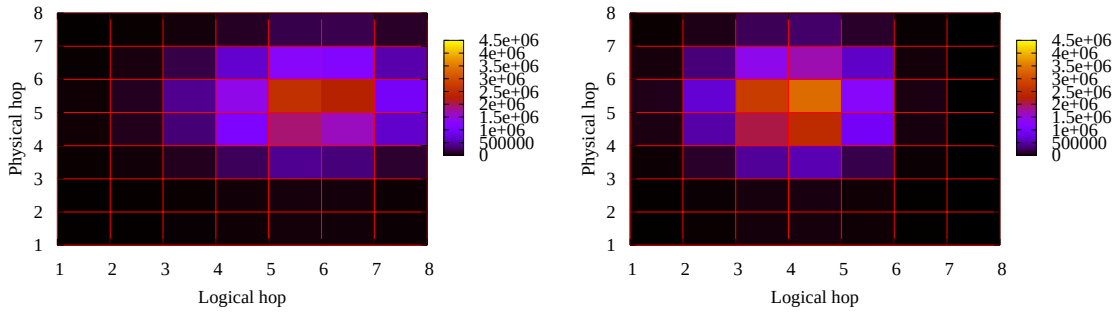
(a) LTM

(b) BA ($m = 3$)



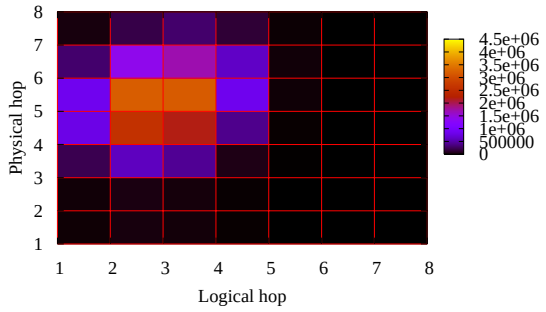
(c) Proposal with rewiring ($m = 3$)

Figure 5.15: Logical hop vs. physical hop (Abilene network)



(a) LTM

(b) BA ($m = 3$)



(c) Proposal with rewiring ($m = 3$)

Figure 5.16: Logical hop vs. physical hop (Sprint network)

Chapter 6

Conclusion

In this chapter, we summarize the discussion in each of the previous chapters and describe future research works.

Chapter 1 mentioned research backgrounds and overviews of our studies. With the growth of computing power and the proliferation of the Internet, media distribution services have widely diffused. However, with the current Internet, the major transport mechanism is still only the best effort service without no guarantees of communication qualities. Furthermore, the media streaming constantly introduces much load into the network compared with other typical applications based on HTTP or FTP. From the viewpoint of users, high levels of QoS, such as high-quality, low-delay, and continuous media play-out, are requested to the streaming services. In addition, the increase of Internet population leads to the diversity of users demands to the media quality and popularity. In this thesis, we focused on the caching mechanisms to accomplish media streaming that can adapt to heterogeneity of user demands and network changes without introducing extra load on the network.

Chapter 2 explained proxy caching mechanisms with a media quality adjustment which could solve the heterogeneity of media qualities requested by users. After

introducing the overview of our system, we proposed three mechanisms: media retrieval mechanism, media prefetching mechanism, and cache replacement mechanism. Through several simulation experiments, we evaluated our proposed mechanisms in terms of required cache buffer size, play-out delay, and media quality. Simulation results showed that our system was effective in suppressing the play-out delay and reducing the required cache buffer size with a media stream of the desired quality. Especially, the degree of user's satisfaction was higher than 0.9 and the play-out delay was less than 8 sec even under a severe environment where the cache buffer size was limited only to 20 Gbits. We further implemented our proposed mechanisms on a real system to verify the practicality and usefulness. Through experiments, it was shown that our implemented system could continuously provide users with a high-quality and low-delay media service in accordance with the network condition.

Chapter 3 described continuous and scalable media streaming on unstructured P2P networks. We first proposed scalable media search methods taking into account temporal order of reference to media data that intended users usually played out a media stream from the beginning to the end. Then, we also proposed media retrieval methods to accomplish continuous media play-out. Through several simulation experiments, we showed that the proposed methods could provide users with continuous media play-out without introducing extra load on the system. Specifically, the FLS method could perform continuous media play-out for popular media streams while reducing the amount of search traffic to $\frac{1}{6}$ compared with full flooding.

Chapter 4 discussed adaptive and robust P2P media streaming. First, inspired by biological systems, we proposed a novel cache replacement algorithm that considered the balance between supply and demand for media streams. Next, to improve the adaptability to network changes and robustness against peer departures, we further proposed an adaptive media retrieval method and a robust media search method.

Simulation results showed that our proposed cache replacement algorithm could accomplish continuous media play-out independent of media popularity and adapt to changes in media popularity. Furthermore, we demonstrated that the adaptive media retrieval method could provide users with more continuous media play-out than the previous method proposed in Chapter 3. In addition, we found that the proposed search method was robust to peer departures.

Chapter 5 focused on the construction of a P2P overlay network to improve the user QoS in terms of search and retrieval time of files. After introducing several related works, we proposed a construction method of a low-diameter and location-aware P2P overlay network where a peer could find many physically-close provider peers. We further proposed a rewiring method to improve the structure of the constructed overlay network. Through evaluations in terms of diameter, physical distance between neighboring peers, and degree distribution of the overlay network, it was shown that our proposed method without rewiring could construct a low-diameter overlay network comparable with the BA model. In addition, we found that the rewiring method contributed to reduction of both the physical distance between neighboring peers and diameter of the overlay network. Especially, the proposed methods can accomplish the same logical reachability as the traditional method while reducing network load to 50%.

From the findings obtained in each chapter, we can conclude that cache-based mechanisms are effective to provide users with continuous media play-out while suppressing the load on networks in both client-server and P2P architectures. Furthermore, proposed adaptive mechanisms can address the heterogeneous user demands, such as access connections to the Internet, and the changes in system conditions, i.e., media popularity, network conditions, and peer departures. We believe that these discussions help to lead toward future media streaming systems.

There are several challenging tasks as future research works. First, we should

consider the combination of our proposed methods. Especially, by applying the construction method of a low-diameter and location-aware P2P overlay network to our P2P media streaming system, we expect that more scalable and continuous media streaming can be accomplished. We also plan to implement our media streaming system on a large-scale and real environment to verify the practicality and usefulness. It is also important to consider the media streaming on mobile and wireless P2P networks. We assume that several modifications and new proposals will be needed to adapt node mobility and unstable wireless connection.

Bibliography

[1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext Transfer Protocol, — HTTP/1.1," *Internet Request for Comments 2068*, January 1997.

[2] J. Postel and J. K. Reynolds, "FTP: File Transfer Protocol," *Internet Request for Comments 959*, Oct. 1985.

[3] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," *Internet Request for Comments 1633*, Dec. 1994.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," *Internet Request for Comments 2475*, Sept. 1998.

[5] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reSerVation Protocol (RSVP) - Version 1 Functional Specification," *Internet Request for Comments 2205*, Sept. 1997.

[6] S. Herzog, "RSVP Extentions for Policy Control," *Internet Request for Comments 2750*, Jan. 2000.

[7] M. Handley, J. Padhye, S. Floyd, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," *Internet draft draft-ietf-tsvwg-tfrc-04.txt*, April 2002.

[8] M. Sasabe, Y. Taniguchi, N. Wakamiya, M. Murata, and H. Miyahara, "Proxy Caching Mechanisms with Quality Adjustment for Video Streaming Services," *IEICE Transactions on Communications Special Issue on Content Delivery Networks*, vol. E86-B, pp. 1849–1858, June 2003.

[9] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Proxy Caching Mechanisms with Video Quality Adjustment," in *Proceedings of SPIE International Symposium ITCom 2001*, vol. 4519, (Denver), pp. 276–284, Aug. 2001.

[10] M. Sasabe, Y. Taniguchi, N. Wakamiya, M. Murata, and H. Miyahara, "Implementation and Evaluation of Proxy Caching Mechanisms with Video Quality Adjustment," in *Proceedings of ITC-CSCC 2002*, vol. 1, (Phuket), pp. 121–124, July 2002.

[11] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Caching Mechanisms for Proxy Cache with Quality Adaptation," *Technical Report of IEICE, (NS2001-53) (in Japanese)*, pp. 31–36, June 2001.

[12] Y. Taniguchi, M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Implementation and Evaluation of Proxy Caching Mechanisms with Video Quality Adjustment," *Technical Report of IEICE, (CQ2002-72) (in Japanese)*, pp. 41–46, July 2002.

[13] R. Rejaie and J. Kangasharju, "Mocha: A Quality Adaptive Multimedia Proxy Cache for Internet Streaming," in *Proceedings of NOSSDAV*, (New York), June 2001.

[14] M. Hofmann, T. S. E. Ng, K. Guo, S. Paul, and H. Zhang, "Caching Techniques for Streaming Multimedia over the Internet," *Technical Report BL011345-990409-04TM*, April 1999.

[15] K.-L. Wu, P. S. Yu, and J. L. Wolf, "Segment-based Proxy Caching of Multimedia Streams," in *Proceedings of the 10th International WWW Conference*, pp. 36–44, 2001.

[16] J. Shudong, B. Azer, and I. Arun, "Accelerating Internet Streaming Media Delivery using Network-Aware Partial Caching," *Technical Report BUCS-TR-2001-023*, October 2001.

[17] M. Reisslein, F. Hartanto, and K. W. Ross, "Interactive Video Streaming with Proxy Servers," *Information Sciences: An International Journal*, December 2001.

[18] B. Wang, S. Sen, M. Adler, and D. Towsley, "Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution," in *Proceedings of IEEE INFOCOM 2002*, (New York), pp. 1726–1735, June 2002.

[19] J. Liu and J. Xu, "Proxy Caching for Media Streaming over the Internet," *IEEE Communications Magazine*, vol. 42, pp. 88–94, Aug. 2004.

[20] B. Shen, S.-J. Lee, and S. Basu, "Caching Strategies In Transcoding-Enabled Proxy Systems For Streaming Media Distribution Networks," *IEEE Transactions on Multimedia*, vol. 6, pp. 375–386, Apr. 2004.

[21] N. Yeadon, F. Garcia, D. Hutchinson, and D. Shepherd, "Filters: QoS Support Mechanisms for Multipeer Communications," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 1245–1262, September 1996.

[22] T. Yamada, N. Wakamiya, M. Murata, and H. Miyahara, "Implementation and Evaluation of Video-Quality Adjustment for Heterogeneous Video Multicast," in *Proceedings of The 8th Asia-Pacific Conference on Communications (APCC 2002)*, (Bandung), pp. 454–457, Sept. 2002.

[23] N. Wakamiya, M. Murata, and H. Miyahara, "Video Streaming Systems with Cooperative Caching Mechanisms," in *Proceedings of IEEE ITCom 2002*, pp. 305–314, July 2002.

[24] N. Wakamiya, M. Murata, and H. Miyahara, "On Proxy-Caching Mechanisms for Cooperative Video Streaming in Heterogeneous Environment," in *Proceedings of IFIP/IEEE MMNS 2002*, pp. 127–139, Oct. 2002.

[25] Y. Taniguchi, N. Wakamiya, and M. Murata, "Implementation and Evaluation of Cooperative Proxy Caching Mechanisms for Video Streaming Services," in *Proceedings of IEEE ITCom 2004*, pp. 288–299, Oct. 2004.

[26] AllCast. available at `http://www.allcast.com`.

[27] vTrails. available at `http://www.vtrails.com`.

[28] Share Cast. available at `http://www.scast.tv`.

[29] D. A. Tran, K. A. Hua, and T. T. Do, "A Peer-to-Peer Architecture for Media Streaming," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 121–133, Jan. 2004.

[30] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient Peer-to-Peer Streaming," *Microsoft Research Technical Report MSR-TR-2003-11*, Mar. 2003.

[31] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," in *Proceedings of ACM SIGCOMM 2002*, pp. 205–217, 2002.

[32] Y. hua Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast," in *Proceedings of ACM SIGMETRICS 2000*, (New York), pp. 1–12, 2000.

[33] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the Internet Using an Overlay Muilticast Architecture," in *Proceedings of ACM SIGCOMM 2001*, (New York), pp. 55–67, 2001.

[34] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava, "On Peer-to-Peer Media Streaming," in *Proceedings of ICDCS2002*, vol. 1, (Vienna), pp. 363–371, July 2002.

[35] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: Peer-to-Peer Media Streaming Using CollectCast," in *Proceedings of ACM Multimedia 2003*, (Berkeley), pp. 45–54, Nov. 2003.

[36] Napster. available at `http://www.napster.com`.

[37] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in *Proceedings of ACM SIGCOMM 2001*, (New York), pp. 149–160, 2001.

[38] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-To-Peer," in *Proceedings of Middleware 2001*, (London), pp. 329–350, 2001.

[39] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," *U.C. Berkeley Technical Report UCB//CSD-01-1141*, Apr. 2001.

[40] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," in *Proceedings of ACM SIGCOMM 2001*, (New York, NY, USA), pp. 161–172, 2001.

[41] Gnutella. available at `http://gnutella.wego.com`.

[42] KaZaA. available at `http://www.kazaa.com`.

[43] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Effective Methods for Scalable and Continuous Media Streaming on Peer-to-Peer Networks," *European Transactions on Telecommunications*, vol. 15, pp. 549–558, Nov. 2004.

[44] M. Sasabe, N. Wakamiya, and M. Murata, "Adaptive and Robust P2P Media Streaming," *WSEAS TRANSACTIONS on COMMUNICATIONS*, vol. 4, pp. 425–430, July 2005.

[45] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Scalable and Continuous Media Streaming on Peer-to-Peer Networks," in *Proceedings of P2P 2003*, (Linköping), pp. 92–99, Sept. 2003.

[46] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Media Streaming on P2P Networks with Bio-inspired Cache Replacement Algorithm," in *Proceedings of Bio-ADIT 2004*, (Lausanne), pp. 380–395, Jan. 2004.

[47] M. Sasabe, N. Wakamiya, and M. Murata, "Adaptive Media Streaming on P2P Networks," in *Proceedings of ATNAC 2004*, (Sydney), pp. 549–556, Dec. 2004.

[48] M. Sasabe, N. Wakamiya, and M. Murata, "Adaptive and Robust P2P Media Streaming," *9th WSEAS International Conference on COMMUNICATIONS*, July 2005.

[49] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Scalable Methods for Media Streaming on Peer-to-Peer Networks," *Technical Report of IEICE, (NS2003-101) (in Japanese)*, pp. 71–76, Sept. 2003.

[50] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, "Cache Replacement Algorithm for P2P Media Streaming," *Technical Report of IEICE, (SB-10-2)*, pp. SE–3–SE–4, Mar. 2004.

[51] R. Schollmeier and G. Schollmeier, "Why Peer-to-Peer (P2P) Does Scale: An Analysis of P2P Traffic Patterns," in *Proceedings of P2P2002*, (Linköping), Sept. 2002.

[52] E. Bonabeau, A. Sobkowski, G. Theraulaz, and J.-L. Deneubourg, "Adaptive Task Allocation Inspired by a Model of Division of Labor in Social Insects," in *Proceedings of BCEC1997*, (Skovde), pp. 36–45, 1997.

[53] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[54] M. Campos, E. Bonabeau, G. Theraulaz, and J.-L. Deneubourg, "Dynamic Scheduling and Division of Labor in Social Insects," *Adaptive Behavior*, vol. 8, no. 2, pp. 83–96, 2000.

[55] M. Sasabe, N. Wakamiya, and M. Murata, "Construction Methods of a Low-Diameter and Location-Aware P2P Network," submitted to *ICDCS 2006*, Nov. 2005.

[56] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, 1999.

[57] Y. Liu, X. Liu, L. Xiao, L. M. Ni, and X. Zhang, "Location-Aware Topology Matching in P2P Systems," in *Proceedings of INFOCOM 2004*, (Hong Kong), Mar. 2004.

[58] A. Beygelzimer, G. Grinstein, R. Linsker, and I. Rish, "Improving Network Robustness," in *Proceedings of ICAC 2004*, pp. 322–323, 2004.

[59] R. Albert and A.-L. Barabási, "Topology of Evolving Networks: Local Events and Universality," *Physical Review Letter*, vol. 85, Dec. 2000.

[60] K. Fukuda, N. Wakamiya, M. Murata, and H. Miyahara, "QoS Mapping between User's Preference and Bandwidth Control for Video Transport," in *Proceedings of IFIP IWQoS '97*, pp. 291–302, May 1997.

[61] UCB/LBNL/VINT, "Network Simulator - ns (version 2)." available at `http://www-mash.cs.berkeley.edu/ns/`.

[62] D. A. Tran, K. A. Hua, and T. Do, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming," in *Proceedings of IEEE INFOCOM2003*, (San Francisco), Mar. 2003.

[63] W. Jeon and K. Nahrstedt, "Peer-to-peer Multimedia Streaming and Caching Service," in *Proceedings of ICME2002*, (Lausanne), Aug. 2002.

[64] C. Man, G. Hasegawa, and M. Murata, "ImTCP: TCP with an Inline Measurement Mechanism for Available Bandwidth," *Computer Communications Special Issue: Monitoring and Measurements of IP Networks*, Sept. 2004.

[65] B. M. Waxman, "Routing of Multipoint Connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, pp. 1617–1622, Dec. 1988.

[66] Zona Reaserch Inc., "The Economic Impacts of Unacceptable Web Site Download Speeds." available at `http://www.webperf.net/info/wp_downloadspeed.pdf`.

[67] R. H. Wouhaybi and A. T. Campbell, "Phenix: Supporting Resilient Low-Diameter Peer-to-Peer Topologies," in *Proceedings of INFOCOM 2004*, (Hong Kong), Mar. 2004.

[68] P. Karbhari, M. Ammar, A. Dhamdhere, H. Raj, G. Riley, and E. len Zegura, "Bootstrapping in Gnutella: A Measurement Study," in *Proceedings of PAM 2004*, (Antibes Juan-les-Pins), Apr. 2004.

[69] J. Gómez-Gardeñes and Y. Moreno, "Local versus Global Knowledge in the Barabási-Albert Scale-Free Network Model," *Physical Review E*, vol. E69, no. 037103, pp. 1–4, 2004.

[70] L. Li, D. Alderson, W. Willinger, and J. Doyle, "A First-Principles Approach to Understanding the Internet's Router-Level Topology," in *Proceedings of ACM SIGCOMM 2004*, (Portland), pp. 3–14, Aug. 2004.

[71] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP Topologies with Rocketfuel," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 2–16, 2004.