Multimedia Parallel Transmission Model for Multi-source Streaming on Peer-to-Peer Networks

Satoshi Itaya, Tomoya Enokido, and Makoto Takizawa

Dept. of Computers and Systems Engineering
Tokyo Denki University, Japan
{itaya, eno, taki}@takilab.k.dendai.ac.jp

Abstract

A peer-to-peer (P2P) network is composed of large number and various types of computers which are cooperating by exchanging data in the Internet. In multimedia streaming applications like music streaming and video on demand (VOD), multimedia data is required to be efficiently delivered to multiple destination processes. In addition, multimedia communication has to satisfy Quality of Service (QoS) requirement, i.e. delay time, bandwidth, and loss ratio. It is significant to efficiently deliver multimedia data to multiple destination processes with QoS requirement. However, each computer may not exchange multimedia data due to the limited computation resource like CPU, memory, and bandwidth of transmission/receipt of data. Thus, each process may not be satisfy QoS requirement even if other process supports enough computation resource. This paper discusses how to efficiently deliver multimedia data on P2P networks so that satisfy QoS requirements.

ピアツーピアネットワーク上でのマルチメディア並列転送モデル

板谷 智史　榎戸 智也　滝沢 誠
東京電機大学大学院 理工学研究科 情報システム工学専攻
{itaya, eno, taki}@takilab.k.dendai.ac.jp

ピアツーピア (P2P) アプリケーション等の大規模分散システムでは, 数百・数千の多種多様なコンピュータがアプリケーションデータの交換により協調動作を行う. 音楽ストリーミングおよびビデオオンデマンド (VOD) のようなマルチメディアストリーミングアプリケーションでは, マルチメディアデータを効率的に複数の宛先プロセスに配送することが要求される. さらに, マルチメディア通信では, 遅延時間, 帯域幅および損失率といったサービス品質 (QoS) を満たすことが必要となる. サービス品質を満たし複数の宛先プロセスに効率的にマルチメディアデータを配送することが重要となる. しかし, 各コンピュータの処理資源には限りがあり, 各コンピュータにおいてデータを送受信できない場合がある. 本論文では, P2P ネットワークにおける効率的なマルチメディアデータ転送方法について議論する.

1. Introduction

In large-scale distributed systems like peer-to-peer (P2P) overlay networks [4, 16, 18, 20, 24], large number of processes are cooperating by exchanging messages. In multimedia streaming applications like music streaming and video on-denamd (VOD) [17], multimedia data is multicast in various types of communication networks like ATM network, Gigabit, 10 Gigabit Ethernet, and wireless networks [1, 2, 8, 9]. Multimedia streaming service [11, 19, 22] is required to be provided for distance learning, e-commerce, home entertainment, and so on. One-to-one/one-to-many types of communication protocols like TCP [15] and RTP [21] are so far developed and underly used for the applications. One-to-one and one-to-many communication protocols to satisfy Quality of Service (QoS) requirements like delay time, bandwidth, and loss ratio are discussed in papers [2, 10, 25].

In the P2P environment [4, 16, 20], large number and various types of computers are interconnected in the Internet. Each computer is equipped with only limited computation resource like CPU, memory, and bandwidth for transmission/receipt of multimedia data in networks. P2P applications are supported by cooperation of multiple peer

application processes which are exchanging multimedia data. Traditional streaming service like RTSP [22] is realized by using one-to-one or one-to-many type of communication, i.e. broadcast and multicast service. There are two approaches to supporting multicast service: network-level (IP) multicast [5] and application-level (*overlay*) multicast [3]. Here, network-level multicast is required to realize multicast networks like MBONE [5]. In the IP multicast, each router has to support multicast functions. On the other hand, in overlay multicast networks, peer processes are not required to support the multicast communication functions.

In the overlay multicast approach, each peer process spends more network resource than IP multicast approach since data transmission path is duplicated. Therefore, it is difficult to support large number of peer processes due to computation and communication overheads on peer-to-peer (P2P) streaming applications. In addition, each peer may not satisfy QoS requirements due to the limited computation resource of each computer. Thus, each peer process has to efficiently support communication of multimedia data on P2P overlay networks. We discuss a high-performance and highly reliable data transmission mechanism for streaming multimedia data on P2P overlay networks. We also discuss how to efficiently deliver packets to the destinations by using multiple source peers. In our protocol, every operational contents peer starts transmitting packets to each leaf peer independently of the other contents peers.

In section 2, we presents system model of multi-source streaming on P2P networks. In section 3, we discuss how to deliver multimedia data from multiple source peers to multiple destination peers.

2. System Model

2.1 P2P environment

A peer-to-peer (P2P) overlay network is composed of large number and various types of computers mainly personal computers which are interconnected in the Internet. A P2P overlay network is realized by cooperation of multiple application processes $AP_1, \ldots, AP_n$ ($n>1$) by taking usage of underlying networks. Application processes are interconnected in *overlay* networks as shown in Figure 1. In the P2P overlay networks, a pair of application processes AP_i and AP_j are interconnected with a logical communication channel $C_{ij} = \langle AP_i, AP_j \rangle$. Multimedia data are delivered from one process to another process through the channel on the P2P overlay network. Multimedia data is decomposed into a sequence *pkt* of packets $\langle t_1, \ldots, t_l \rangle$ ($l \geq 1$). A packet is a unit of data transmission in networks. A sequence *pkt* of packets are sent to destination peer processes by using underlying network protocols like UDP [14] and TCP [15].

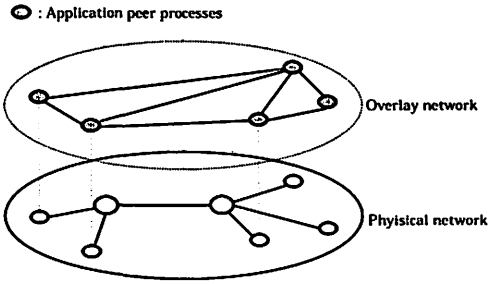

Figure 1. Overlay network.

2.2 Multi-source streaming

In P2P applications like video on-denmand (VOD), multiple peer processes (abbreviated *peers*) are cooperating by exchanging multimedia data with other peer processes. In this paper, we take an application for delivering multimedia contents to one or more than one peer process on request of the processes. There are two types of peers, *contents* peers and *leaf* peers. A contents peer receives a request of some content from a leaf peer and then starts transmitting a sequence of packets of the multimedia content to the leaf peer. In traditional model, one contents peer supports multiple leaf peers and transmits packets of the content to each leaf peer asynchronously with the other leaf peers. Each leaf peer receives a sequence of the packets from one contents peer. A contents peer may be performance bottleneck and single point of failure.

In order to realize the higher reliability and throughput, we take a novel approach, using redundant contents peers. Let denote a set of multiple contents peers, i.e. **CP** = $\{CP_1, \ldots, CP_n\}$ ($n \geq 1$). Let **LP** be a set of leaf peers $LP_1, \ldots, LP_m$ ($m \geq 1$) which use a content in the contents peers. Each leaf peer LP_i issues a request of the content C independently of the other leaf peers. On receipt of requests of a content C from multiple leaf peers, a contents peer CP_i multicasts a sequence of packets of the content C to the leaf peers in a P2P overlay network. Multimedia data is delivered from multiple contents peer to multiple leaf peers via multiple paths. Each leaf peer receives packets from one contents peer and another leaf peer may receive packets from another contents peer. The overhead of the contents peer is distributed to multiple peers. This is a traditional approach [Figure 2].

We take a new approach named multi-source streaming approach. Here, each client receives packets of a multimedia content C from multiple contents peers while each contents peer sends a packet to multiple client peers.

3. Parallel Transmission Procedure

3.1 Transmission

We discuss how multiple contents peers deliver packets of a content C to each client peer which issues a request

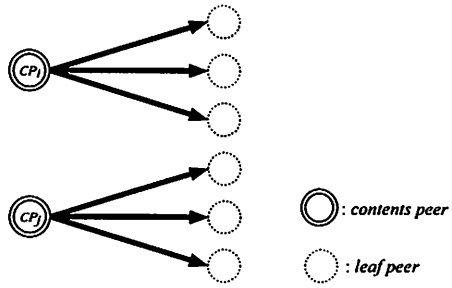

Figure 2. Traditional approach.

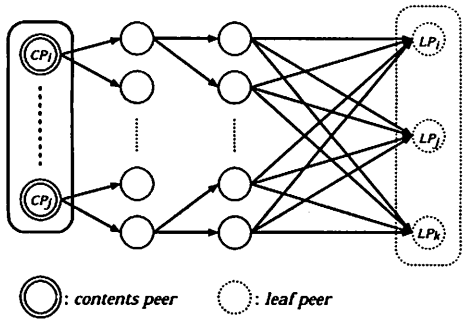

Figure 3. Multi-source streaming.

of the content C. We first show the overview of the transmission procedure as follows:

1. A leaf peer LP_j sends a request to contents peer $CP_1, \ldots, CP_n$ in **CP**.
2. Multiple contents peers $CP_1, \ldots, CP_n$ ($n \geq 1$) multicast different packets to the leaf peer LP_j.
3. The leaf peer LP_j receives packets through multiple channels with each CP_i of the contents peers.

Multiple channels with each contents peer are classified into one *main* channel and *sub* channels. If each contents peer CP_i sends a same sequence of packets to each leaf peer LP_j, LP_j has to receive many redundant packets and may be performance bottleneck due to the heavy traffic. In our approach, each contents peer CP_i sends a leaf peer LP_j packets different from every other peer CP_k ($k \neq i$). Suppose that pkt is a sequence $\langle t_1, \ldots, t_l \rangle$ of packets of the content C to be delivered to the leaf peer LP_j. The content C is replicated in multiple contents peers CP_1, ..., CP_n. **CP** is a collection $\{CP_1, \ldots, CP_n\}$ of the contents peers which have the replicates of the content C. There are m leaf peers $LP_1, \ldots, LP_m$, **LP** = $\{LP_1, \ldots, LP_m\}$.

A contents peer CP_i sends a subsequence pkt_{ij} of the packets to each leaf peer LP_j ($pkt_{ij} \subseteq pkt$). The subsequence pkt_{ij} of the packet sequence pkt is composed of packets $\{ t_h \mid h = i + n{\cdot}d$ for $d = 0,1,\ldots\}$. For example, suppose a packet sequence $pkt = \langle t_1, t_2, t_3, t_4, t_5, t_6, t_7 \rangle$ is obtained from a multimedia content C and there are three contents peers CP_1, CP_2, and CP_3 where the content C is stored. A leaf peer LP_1 first sends a request of the content C to the contents peers. Here, the contents peer CP_1 sends a subsequence pkt_{11} of the packets $\langle t_1, t_4, t_7 \rangle$, CP_2 sends a subsequence $pkt_{21} = \langle t_2, t_5 \rangle$, and CP_3 sends a subsequence $pkt_{31} = \langle t3, t_6 \rangle$ to the leaf peer LP_1. The leaf peer LP_1 receives the subsequences pkt_{11}, pkt_{21}, and pkt_{31} from the contents peers CP_1, CP_2, and CP_3, respectively. Then, the leaf peer LP_j obtains the packet sequence pkt from the subsequences pkt_{11}, pkt_{21}, and pkt_{31}. Thus, the packet sequence pkt is partitioned into n subsequences $pkt_{1j}, \ldots, pkt_{nj}$ for a leaf peer LP_j, where each subsequence pkt_{ij} is transmitted to LP_j by a contents peer CP_i.

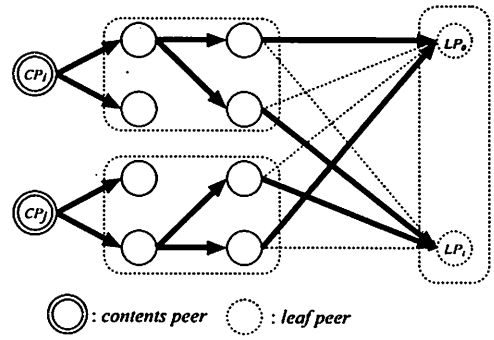

Figure 4. Transmission.

On receipt of a request of a content C from a leaf peer LP_j, every contents peer CP_i generates a sequence pkt of packets from the multimedia content C. Packets in the packet sequence $pkt = \langle t_1, \ldots, t_l \rangle$ are enqueued into a local queue (LQ_i) of CP_i ($i = 1, \ldots, n$). Packets in the local queue LQ_i are dequeued and enqueued into transmission queues XQ_{i1}, ..., XQ_{in}. Each packet t_k is dequeued from the local queue LQ_i. $h = \mathrm{mod}(k-1, n) + 1$ for the number n of contents peers. The packet t_k is enqueued into a transmission queue XQ_{ih}. On receipt of a request from a leaf peer LP_j, a function $Qid(i, j)$ is executed and returns some number h. The function Qid has the following properties.

- $Qid(i, j) \neq Qid(i', j)$ if $i \neq i'$.

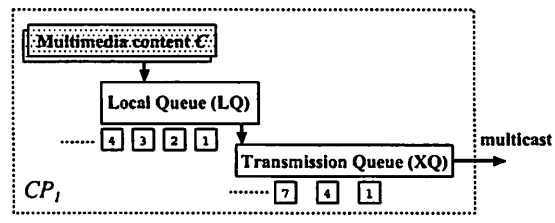

Figure 5. Decomposition of a multimedia content C into packets.

For example, $Qid(j, i)$ is realized as $(j + i) \bmod n + 1$. Then, the contents peer CP_i transmits packets to the

leaf peer LP_j from the transmission queue XQ_{ik} here $k = Qid(i, j)$ [Figure 6]. The contents peer CP_i transmits another subsequence in the queue $XQ_{ik'}$ to another leaf peer $LP_{j'}$ when $k' = Qid(i, j')$.

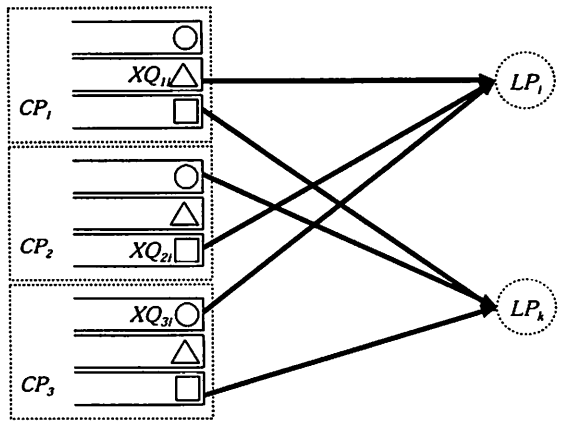

Figure 6. Transmission queues.

3.2 Centralized cooperation of contents peers

Suppose a leaf peer LP_j requests the contents peers $CP_1, \ldots, CP_n$ to transmit a multimedia content C. As presented before, each contents peer CP_i starts transmitting a subsequence pkt_{ij} to the leaf peer LP_j ($i = 1, \ldots, n$). Questions are to which contents peer the leaf peer LP_j sends the request of the content C and how all the contents peers start transmitting packets to the leaf peer LP_j. There is two approaches, *centralized* and *distributed* ones. In the centralized approach, the leaf peer LP_j sends a request to one of the contents peers, say CP_1 which is a controller. The controller peer CP_1 coordinates the synchronization of the transmission of packets among all the contents peers [Figure 7]. For example, the contents peers starts transmitting packets by using the two-phase commitment (2PC) protocol [6,7,23] as follows:

1. The controller peer CP_1 sends a *prepare* message to all the other contents peers $CP_2, \ldots, CP_n$.
2. On receipt of the *prepare* message from the controller peer CP_1, a contents peer CP_i prepares the transmission of the content C, i.e. packets are enqueued to the transmission queue XQ_{ik_i} where $k_i = Qid(i, j)$ ($i = 2, \ldots, n$). Then, CP_i sends an *acknowledgment* (*ACK*) message to the controller CP_1.
3. If the controller peer CP_1 receives *ACK* messages from all the contents peers $CP_2, \ldots, CP_n$, the controller peer CP_1 sends a *start* message to all the contents peers $CP_2, \ldots, CP_n$. The controller CP_1 starts transmitting packets from the transmission queue XQ_{1k_1} to the leaf peer LP_j where $k_1 = Qid(1, j)$.
4. On receipt of the *start* message from the controller peer CP_1, the contents peer CP_i starts transmitting packets from XQ_{ik_i} to the leaf peer LP_j.

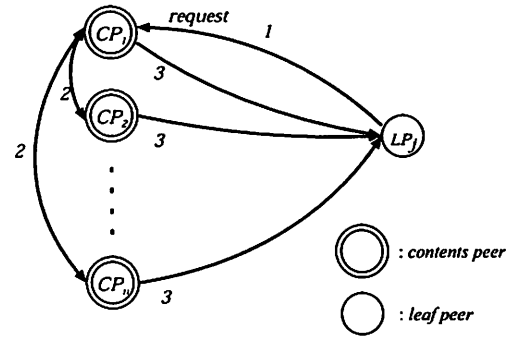

Figure 7. Centralized approach.

3.3 Asynchronous cooperation of contents peers

On the other hand, there is no centralized controller in the distributed approach. A leaf peer LP_j sends a request to all or some of the contents peers $CP_1, \ldots, CP_n$. Here, there are two ways to start the transmission of packets at the contents peers; *synchronous* and *asynchronous* one. In the synchronous transmission, all the contents peers are synchronized to simultaneously start the transmission of packets. Protocols similar to the two-phase commitment (2PC) protocol can be used to synchronize all the contents peers. It takes time to exchange packets to synchronize all the contents peers, i.e. at least three rounds.

In the asynchronous transmission, each contents peer starts transmitting the packets independently of the other contents peers. Here, some contents peer, say CP_k may not be ready while another contents peer CP_i starts transmitting packets. A contents peer CP_i is referred to as *operational* if CP_i is transmitting packets. Each contents peer CP_i manipulates a sequence number variable SQ. SQ shows a sequence number of a packet which has been most recently transmitted by CP_i. A variable SQ_j ($j = 1, \ldots, n$) is also manipulated, which denotes the sequence number of a packet which the contents peer CP_i has most recently received from another one CP_j. Initially, each variable SQ_j is zero. In addition, CP_i manipulates a matrix of sequence number variables $MVQ = \{MVQ_{ij} \mid i, j = 1, \ldots, n\}$ where each element MVQ_{ij} is initially 0. Each contents peer exchanges the sequence vector $VSQ = \langle SQ_1, \ldots, SQ_n \rangle$ with the other processes. On receipt of a packet m with a sequence number $m.SQ$ and a vector $m.SQ = \langle m.SQ_1, \ldots, m.SQ_n \rangle$ from a contents peer CP_j, the variables are manipulated in the contents peer CP_i as follows:

[Receipt procedure] CP_i receives a packet m,

1. $SQ_j := m.SQ$.
2. $MVQ_{jk} := \max(MVQ_{jk}, m.SQ_k)$ ($k = 1, \ldots, n$).

[Sending procedure]

1. On sending a packet p, $SQ := p.SQ$ and $p.VSQ = \langle SQ_1, \ldots, SQ_n \rangle$.
2. Send the packet p.

As discussed in papers [12, 13], the contents peer CP_i knows that every contents peer CP_j has transmitted a packet whose sequence number SQ is equal to or smaller than $\min(M_1, \ldots, M_n)$, where each M_k satisfies the followings:

1. $M_k = MVQ_{jk}$ if $MVQ_{jk} \neq 0$ otherwise $\top$.
2. $M_1 = \cdots = M_n = 0$ if $MVQ_{j1} = \cdots = MVQ_{jn} = 0$.

$MVQ_{jk} = 0$ means that no contents peer knows that the contents peer CP_k is operational. A value $\top$ shows the maximum value. Here, let MSQ_j be the maximum sequence number of such a packet from the contents peer CP_j that CP_i has received packets from CP_j, i.e. CP_j is operational. Let $\mathbf{CCP}_i$ be a subset of contents peers which CP_i knows to be operational ($\mathbf{CPP}_i \subseteq \mathbf{CP}$). The contents peers are ordered in the peer number in $\mathbf{CCP}_i$. Here, $No(CP_i)$ shows the order of the contents peer CP_i in $\mathbf{CCP}_i$.

First, the contents peer CP_i receives a request of a content C from a leaf peer LP_j. Here, $\mathbf{CPP}_i = \phi$. Hence, the contents peer CP_i starts transmitting a sequence pkt of packets, i.e. transmits the first packet t_1, the second packet t_2, ... The contents peer CP_i distributes the sequence number vector to all the contents peers and receives from other peers. A packet which carries the sequence number is referred to as *control* packet. If the contents peer CP_i sends a control packet to the other contents peers each time CP_i sends a packet of the content, the communication overhead is increased. In our protocol, the contents peer CP_i sends a control packet each time the contents peer CP_i sends some number of packets to reduce the communication overhead. After exchanging control packets among the contents peers, $\mathbf{CCP}_i \neq \phi$. Here, the contents peer CP_i sends a subsequence of the packet sequence pkt. Each pair of contents peers CP_i and CP_k transmit different subsequences pkt_{ij} and pkt_{kj} to the leaf peer LP_j, $pkt_{ij} \neq pkt_{kj}$. The contents peer CP_i transmits a packet m where $\mathrm{mod}(m.SQ\text{-}1, |\mathbf{CPP}_i|) = No(CP_i)$ and $m.SQ \leq \min\{MSQ_k \mid CP_k \in \mathbf{CCP}_i\}$.

After some contents peers start transmitting packets to the leaf peer LP_j, another contents peer CP_i would start transmitting packets. If the contents peer CP_i had not received any sequence number vector, the contents peer CP_i starts transmitting a sequence pkt of packets to LP_j and distributes the sequence number vector. In the meanwhile, the contents peer CP_i receives the sequence number vectors from other contents peers. On receipt of a packet m with the vectors from CP_k, the contents peer CP_i obtains the sequence number $SQ_k := m.SQ$. Here, if $SQ < m.SQ$, the contents peer CP_i skips packets in pkt and then sends a packet m where $m.SQ = SQ_k + 1$. In the meanwhile, the contents peer CP_i takes the transmission way as discussed here.

3.4 Redundant transmission

Some contents peer may be faulty and packets may be lost. In order to be tolerant of the faults, the contents peers redundantly transmit packets to each leaf peer. For some number of packets $t_1, \ldots, t_k$, one parity packet pt is created. Even if one packet of $(k+1)$ packets $t_1, \ldots, t_k$, and pt is lost, the packet lost can be recovered by the other k packets. k is decided by the number of operational contents peers, i.e. $k < |\mathbf{CCP}_i|$. One parity packet is inserted every k packets. Here, let t_{se} show a parity packet of packets $t_s, t_{s+1}, \ldots, t_e$. Figure 8 shows three contents peers CP_1, CP_2, and CP_3 transmit packets to the leaf peer LP_j.

For example, even if a packet t_3 is lost by the leaf peer LP_j, the packet t_3 can be recovered by the normal packet t_4 and the parity packet t_{34}. Even if CP_3 is faulty, the leaf peer LP_j can receive the packet sequence $t_1, t_2, \ldots$ of the content C.

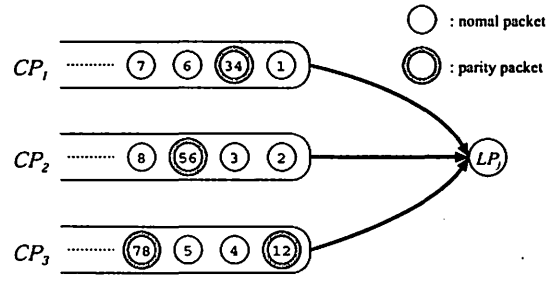

Figure 8. Redundant transmission.

4. Concluding Remarks

The paper discuss the multi-source streaming approach to transmit multimedia contents from multiple contents peers to leaf peers. We discuss the asynchronous multi-source streaming protocols. Here, each contents peers can start transmitting packets independently of the other contents peers. While transmitting packets and exchanging control information, every operational peer sends different subsequence to the leaf peer.

References

[1] 10 Gigabit Ethernet Alliance. http://www.10gea.org/.

[2] ATM Forum. Traffic Management Specification Version4.0. 1996.

[3] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. *Proc. of ACM SIGCOMM*, 2001.

[4] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. *Proc. of the Workshop on Design Issues in Anonymity and Unobservability*, pages 311–320, 2000.

[5] H. Eriksson. MBONE: the multicast backbone. *Communications of the ACM*, 37(8):54–60, 1994.
[6] J. Gray. Database: Principles, Programming, and Performance (Second Edition). *Morgan Kaufmann*, 2001.
[7] J. Gray and A. Reuter. Transaction Processing : Concepts and Techniques. *Morgan Kaufmann*, 1993.
[8] IEEE Standards Association. Gigabit Ethernet. *IEEE Standard 802.3z*, 1998.
[9] IEEE Standards Association. 10 Gigabit Ethernet. *IEEE Standard 802.3ae*, 2002.
[10] S. Itaya, T. Tojo, T. Enokido, M. Rozeta, and M. Takizawa. QoS-Based Synchronous/Asynchronous Data Transmission Model in Group Communication. *Proc. of IEEE the 18th International Conference on Advanced Information Networking and Applications (AINA-2004)*, 1:35–40, 2004.
[11] Microsoft Windows Media Technology. http://www. microsoft.com/windows/windowsmedia/.
[12] A. Nakamura and M. Takizawa. Priority-Based Total and Semi-Total Ordering Broadcast Protocols. *Proc. of IEEE the 12th International Conference on Distributed Computing Systems (ICDCS-12)*, pages 178–185, 1992.
[13] A. Nakamura and M. Takizawa. Causally Ordering Broadcast Protocol. *Proc. of IEEE the 14th International Conference on Distributed Computing S ystems (ICDCS-14)*, pages 48–55, 1994.
[14] J. Postel. User Datagram Protocol. *RFC768*, 1980.
[15] J. Postel. Transmission Control Protocol. *RFC793*, 1981.
[16] Project JXTA. http://www.jxta.org/.
[17] P. V. Rangan, H. M. Vin, and S. Ramanathan. Designing an On-Demand Multimedia Service. *IEEE Communications Magazine*, 30(7):56–65, 1992.
[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content-Addressable Network. *Proc. of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 161–172, 2001.
[19] Real Networks. http://www.realnetworks.com/.
[20] M. Ripeanu. Peer-to-Peer architecture case study: Gnutella network. *Proc. of International Conference on Peer-to-Peer Computing (P2P2001)*, pages 99–100, 2001.
[21] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real Time Applications. *RFC1889*, 1996.
[22] H. Schulzrinne, A. Rao, and R. Lanphier. Real-Time Streaming Protocol (RTSP). *RFC2326*, 1998.
[23] D. Skeen. Nonblocking Commitment Protocols. *Proc. of ACM SIGMOD*, pages 133–147, 1982.
[24] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. *IEEE/ACM Transactions on Networking (TON)*, 11(1):17–32, 2003.
[25] T. Tojo, T. Enokido, and M. Takizawa. Notification-Based QoS Control Protocol for Multimedia Group Communication in High-Speed Networks. *Proc. of IEEE the 24th International Conference on Distributed Computing Systems (ICDCS-2004)*, pages 644–651, 2004.