A HYBRID VIDEO STREAMING SCHEME ON HIERARCHICAL P2P NETWORKS

Shinsuke Suetsugu, Naoki Wakamiya and Masayuki Murata
Graduate School of Information Science and Technology, Osaka University
1–5 Yamadaoka, Suita-shi, Osaka 565-0871, Japan
email: {suetugu,wakamiya,murata}@ist.osaka-u.ac.jp
Koichi Konishi and Kunihiro Taniguchi
Internet Systems Research Labs., NEC Corporation
1753 Shimonumabe, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8666, Japan
email: k-konishi@cq.jp.nec.com, k-taniguchi@da.jp.nec.com

ABSTRACT

Video streaming based on peer-to-peer networking technology has received much attention from researchers and developers. In this paper, we propose a new scheme for scalable and robust video streaming on P2P networks. Our scheme constructs hierarchical distribution trees from peers and considers the physical structure of underlying hierarchically organized networks, such as an enterprise network. A centric server manages the construction of a higher-level tree among branches, but lower-level trees are built through local communication among peers. Peers recover from faults by themselves, using partial information they have. Through simulation and practical experiments, we verified that our scheme can provide thousands or ten thousands of users with continuous video streaming services on an unreliable P2P network.

KEY WORDS
peer-to-peer, streaming, multicast

1 Introduction

Video streaming is a promising application that has been widely diffused. However, for providing a considerable number of users with streaming service of satisfactory quality, a content provider must provide such infrastructure as broader bandwidth, high-performance streaming servers and storage, and mirror or proxy servers located close to the users. IP multicast is an attractive technology that reduces the bandwidth requirement, but it is not readily available in the current Internet market due to tardy deployment.

Recently, users and service providers are bestowing much attention on peer-to-peer networking technology and its applications. The concept of P2P itself is not new, but because of rapidly growing access-link bandwidth with low or flat-rate service, and inexpensive, but high-performance computers, it now has great potential for changing communication paradigms of computer networks. Hosts called peers who participate in P2P networks communicate with each other to exchange information, cooperatively perform tasks, and build a fully-distributed information system without mediation from servers.

In a P2P network, each peer behaves as a server, a client, and a router. Thus, it is natural to employ P2P networking technology to realize so-called application-level multicasting where peers relay received data to other peers and organize multicast trees. There has been much research on such application or user-level multicasting such as [1, 2, 3, 4]. They constructed multicast trees on an overlay network of end systems or peers in P2P networks. To avoid the waste of bandwidth, they considered characteristics of underlying physical networks consisting of physical links, routers, and hosts. Complete or partial information of physical networks is given in advance or obtained through active or passive measurements.

In this paper, we also consider another scheme for constructing distribution trees in an enterprise network for a video streaming service. In our mechanism, we construct a hierarchical distribution tree in accordance with the structure of an underlying physical network. More specifically, we assume an enterprise network or a network of a large organization that usually has a hierarchical structure of branch networks and division networks. An enterprise network consists of branch networks connected to each other through dedicated channels provided by VPN services. A branch network further consists of division networks in which the propagation delay among any arbitrary two peers is small and sufficient bandwidth is available. By assuming such structured networks, our scheme can quickly build distribution trees without time-consuming measurement and complicated routing control. A server has the minimum knowledge about branches and divisions given by a service administrator in advance. Tree construction is performed in a distributed and scalable way where no servers or peers need to have complete knowledge of a distribution tree. When a peer leaves a tree or a router and a link fails, a child peer of a failed peer or peers behind a failed network facility recover the tree in a distributed way based on information they obtained in joining the tree. We propose a scheduling algorithm for continuous video play-out, a tree construction mechanism for physi-

cally suitable distribution trees, and a fault recovery mechanism that maintains video streaming service on unreliable P2P networks.

The rest of the paper is organized as follows. We describe our proposal in section 2. We first evaluate our scheme through simulation experiment in section 3. Then we show results of practical experiments in section 4. Section 5 summarizes the paper and describes some future works.

2 A hybrid video streaming scheme

In this section we describe our algorithm and mechanisms for video streaming services on enterprise P2P networks. Our system consists of a video streaming server (ORG) from which video streams are distributed, a scheduling server (SS) which schedules video distribution using pyramid broadcasting protocol (PB) [5], a global tree server (GTS) which manages video distribution trees, and peers. We note that one equipment can play more than one part among these; ORG, SS, and GTS. Peers constitute a distribution tree on a segment-by-segment and slot-by-slot basis. The root of a distribution tree is ORG. Internal nodes and leaves of a distribution tree are peers. Edges are logical links; specifically, TCP sessions in our system are established between pairs of peers.

2.1 Overview of a hybrid streaming scheme

A peer who wants to watch a video stream first sends a request message called a *schedule request* to SS. Our scheme employs the pyramid broadcasting protocol (PB) for bandwidth-efficient video distribution [5]. When the duration of the first segment is W [sec], a video stream is divided into segments such that the duration of the i-th segment becomes $\alpha^{i-1}W$. Each segment is repeatedly broadcast on a dedicated channel at α times the rate of encoding rate b. An example of $\alpha = 2$ is shown in Fig. 1. Each segment duration is called a slot. Among schemes for near-on-demand video distribution, we adopt PB for its simplicity, but our mechanism can be applied to systems with other schemes such as Skyscraper [6]. For every segment, SS determines a time slot during which a peer receives it. Details of our scheduling algorithm will be given in subsection 2.2.

To participate in a distribution tree and receive the first segment at the specified slot, a peer sends a request message called a *participation request* to GTS. For the following segments, type and destination of requests differ among peers depending on their roles in the tree of the preceding segment. Distribution trees are hierarchically constructed from peers in segment-by-segment and slot-by-slot fashions, considering the physical structure of the underlying networks. The root of a tree is ORG and internal nodes and leaves are peers that are assigned to the same slot of the same segment. The mechanism for tree construction will be given in Subsection 2.3.

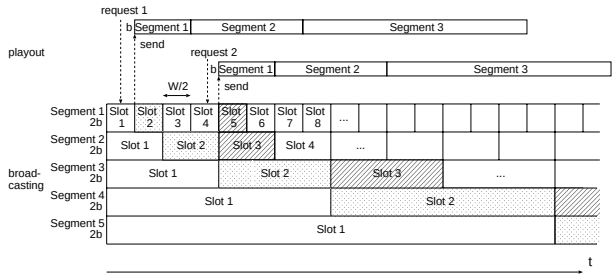

Figure 1. Examples of slot assignment

Once a tree is constructed, it should be maintained for the duration of the slot so that all participating peers can successfully receive the segment. However, since leaves and internal nodes are peers which are inherently unreliable, they often give up their roles in a distribution tree during video distribution. The video distribution is disturbed, when a peer accidentally leaves a distribution tree due to system failure or user interruptions, a link breaks, or a router halts. We call this *fault*. When such faults occur, a fault recovery mechanism described in Subsection 2.4 is conducted to recover the video distribution.

2.2 Scheduling algorithm

For the first segment, a slot which starts the earliest is assigned to a peer in PB. However, since a video distribution tree for the slot is constructed through peer-to-peer communications, it takes some time before a tree is prepared for video distribution. Thus, if SS assigns a slot that starts immediately after the reception of a schedule request, a peer won't be prepared in time, and so it fails to receive the first segment. We define *reception time* for each slot of the first segment during which SS accepts schedule requests for the slot. The duration of each reception time is W/α seconds long and begins C seconds ahead of the slot. C, called *reserved time*, is determined long enough for a peer whose schedule request is accepted at the end of the reception time to participate successfully in a corresponding distribution tree before the beginning of the slot.

For succeeding segments, according to PB, slots are assigned to a peer so that it receives segments one-by-one from the beginning of the stream to the end. For example, a peer, whose schedule request denoted with *request 1* in Fig. 1, is assigned those slots indicated with dotted rectangles. However, for high-quality and continuous video streaming on unreliable P2P networks, SS assigns two overlapping slots of the first and second segments to a peer when the beginning of those slots are the same. For example, a peer with *request 2* is assigned slots 5, 4, 3, 3, and 3 for segments 1, 2, 3, 4, and 5, respectively, in an original PB. On the other hand, in our mechanism, slots 5, 3, 3, 3, and 3 are assigned to the peer. A peer with original PB cannot provide fault lasting for a long time on the first and

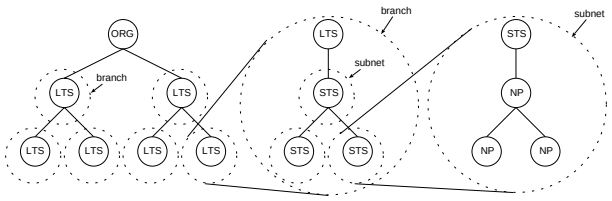

Figure 2. Hierarchical Distribution Tree

second segments, whereas a peer using our algorithm can deposit much data at the early stage of the video distribution, sacrificing bandwidth.

To join a distribution tree and receive segment i on slot s_i which starts at time t_i, a peer must send a participation request during a period from $t_i - W/\alpha - C$ to $t_i - C$. As a response to a schedule request, SS makes a list of slot identifiers s_i on which a peer must receive segment i and their corresponding time $t_i - W/\alpha - C$ from which a peer must issue a corresponding participation request.

2.3 Tree construction mechanism

A video distribution tree is hierarchically constructed to consider the physical structure of the enterprise and organized networks as shown in Fig. 2. An enterprise network consists of medium networks each of which corresponds to a branch office of a corporation, a faculty of a university, or a stronghold of an organization. A branch network is made of further smaller sub-networks of divisions, departments, and groups. In constructing a logical distribution tree for a slot of a segment, for each branch a representative peer called a Local Tree Server (LTS) is chosen among peers which belong to the branch and are assigned to the slot. An inter-branch distribution tree is constructed among LTSs rooted at ORG. A branch network consists of one LTS and subnets. For each subnet, a representative peer called a Subnet Tree Server (STS) is chosen among peers that are assigned to the slot in the corresponding sub-network, except for an LTS. One STS among all of the STSs in a branch is connected to the LTS as a direct child. We call a peer connected to a parent by a different type a direct child of the parent. An inter-subnet tree is organized by STSs whose roots are a direct child of the LTS. A subnet further consists of one STS and normal peers (NPs). Peers which are neither LTS nor STS are called NPs. The root of an inter-NP tree is an NP, which is a direct child of the STS of the subnet. Independent of types as LTS, STS and NP, peers and ORG have a limitation on the number of children called the *fanout*, in accordance with processing capability, the performance of the network interface, and available bandwidth.

In our scheme, a peer first sends a schedule request to SS and receives a schedule. It immediately sends a participation request to GTS to join a distribution tree and receives the first segment. A peer joins a distribution tree by a recursive mechanism such as in TBCP [7] and HMTP [8]. After receiving a request, GTS determines the peer type among LTS, STS, and NP and introduces a temporary parent. For this purpose, GTS maintains an *address table* for each slot of each segment. An address table consists of the information of branches and subnets. One simple and easy way to identify a branch and subnet to which a requesting peer belongs is to use an IP address. In such cases, for each branch, a name, a network address of the branch, an IP address of the LTS, and information about subnets in the branch are maintained. For each subnet, a network address of the subnet and an IP address of the STS are maintained. Branch names, network addresses of branches, and network addresses of subnets are given by a system administrator beforehand. We can easily obtain those information from a network management system, e.g., LDAP server. IP addresses of LTSs and STSs are initially set to 0.0.0.0. GTS knows LTS and STS, but does not have any further detailed information such as the topology of trees.

If an LTS is not yet assigned to a branch to which a requesting peer belongs, GTS appoints the peer to be the LTS. GTS updates an address table and informs the peer of its peer type, i.e., LTS, and the IP address of ORG as a temporary parent. If an LTS already exists, but there is no STS in the peer's subnet, the peer becomes an STS. In this case, GTS updates an address table and informs the peer of its peer type, i.e., STS, and IP address of the LTS as a temporary parent. Otherwise, GTS appoints the peer as NP and introduces an STS as a temporary parent.

On receiving a response from GTS, a peer tries to connect to an informed temporary parent by sending a *connection request*. On receiving a request, a temporary parent decides whether to accept the peer as its child in accordance with peer types and the number of children. All peers participating in a distribution tree maintain a list of children. Consider the case where peer A receives a request from peer B. Peer A compares its type with peer B's type. If they are the same, they belong to a tree of the same level. Peer A accepts peer B as its child if the number of its children is smaller than the fanout. If their types are the same but the number of children already satisfies the fanout, peer A introduces one of its children as a new temporary parent to peer B. A new temporary parent is chosen among children in a round-robin fashion. As a result, a distribution tree is constructed in breath-first order, and the segment transmission delay from ORG to peers can minimalized. If their types are different, it means that peer A is either an LTS or an STS and the requesting peer is either an STS or an NP, respectively. If peer A does not have a direct child in its lower-level tree, it accepts peer B as a direct child. If there already exists a direct child, peer A introduces the direct child to peer B as a new temporary parent. If peer B is LTS, ORG is a temporary parent, and a decision is made based on the number of children and the fanout. Eventually, a requesting peer can successfully be connected to a participating peer in a distribution tree through being intro-

duced to temporary parents. When it participates in a tree, it has a list of ancestors or temporary parents to which it tried to connect.

To join a distribution tree and receive the succeeding segment i ($i \geq 2$) on slot s_i, a peer sends a request, indicating a video identifier, a segment identifier, a slot identifier, and its own identifier, i.e., IP address, during $t_i - W/\alpha - C$ and $t_i - C$. The destination and type of a request depends on the type of a peer in a distribution tree of the preceding segment $i-1$. The order of joining distribution trees follows the order of the segment identifier. Thus, even if an assigned schedule indicates that a peer has to receive both of the first and second segments at the same time, it first finishes joining a tree for the first segment before sending a request for the second segment.

A peer that was NP in the tree of the preceding segment $i-1$ considers STS in the tree of segment $i-1$ as a temporary parent. When a former STS also plays the role of STS in the current segment, an inter-NP tree is constructed through communications within a subnet. It sends connection request to the STS at the random instant from $t_i - W/\alpha - C$ to $t_i - C$, except for the case when $t_i - W/\alpha - C$ has passed where a peer sends a connection request immediately. If a former STS is demoted to NP in a tree of segment i, it introduces a new STS to a requesting peer. If a former STS is promoted to an LTS in a current tree, it tells a requesting peer to send a participation request to GTS recognizing a new STS. If a peer was either of LTS or STS in a tree for segment $i-1$, it sends a participation request to GTS at $t_i - W/\alpha - C$ or immediately if the instant has passed; it joins a tree as fast as possible to answer requests from NPs.

2.4 Fault recovery mechanism

When a peer cannot communicate with a temporary parent in joining a distribution tree, a peer participating in a tree leaves during tree construction or segment distribution, or a network facility such as a link and a router fails, the video distribution is disturbed. When such faults occur, a distribution tree is re-constructed by the fault recovery mechanism described in this subsection. We assume that those faults occur without any notification. They are detected by requesting peers, child peers, and parent peers. We call a peer which becomes unavailable a failed peer. A parent peer only removes a failed peer from its children list. Peers who are trying to connect to a failed peer and the children of a failed peer conduct the fault recovery. As a result of fault recovery, a peer which detected a fault would be promoted from NP to STS or from STS to LTS. It is also detected as a fault by its children.

If the types of a peer which detects faults and that of a failed peer are the same, the peer sends a connection request to its grandparent found in an ancestor list. The failed peer is removed from the list. Then, the same process as in the tree construction is performed, and the peer can join the tree again. In cases where the grandparent has already left the tree, the peer removes the grandparent from the list. Then it tries to send a connection request to a peer newly located at the bottom of the list, which used to be a great-grandparent before the fault. This visiting ancestors procedure is repeated until it can successfully be accepted as a child or until it is introduced to a new temporary parent. If an ancestor list becomes empty during fault recovery, a peer sends a participation request to GTS. It also sends notification to inform GTS of the failure of the LTS or the STS, which is located at the top of the list, so that GTS can update an address table. On the contrary, different types means that a failed peer is LTS or STS. A peer sends a participation request and a failure notification of the server to GTS.

When a peer successfully re-joins a distribution tree, it resumes receiving the segment from a new parent. Every peer deposits segment data for a while after it finishes watching the segment so that afterward it can provide a new child with the requested portion of the segment.

3 Simulation experiments

We evaluated our scheme from a viewpoint of *load of GTS*, *load of peers*, *initial waiting time*, *fault recovery time*, and *freeze time*. The load of GTS is the number of participation requests and notifications of failure of LTS and STS received by GTS per second. The load of the peers is the maximum among the number of connection requests that every peer, independent of its type, receives per second. The time from when a peer sends a schedule request to SS to when it begins to receive the first segment is called the initial waiting time. Fault recovery time corresponds to the time from when a peer detects a fault to when it is connected to a new parent peer. The total periods of starvation of buffer is called freeze time.

3.1 Simulation Conditions

In the simulation experiments in this section, we assumed that there were five branches in an enterprise network and that each branch had five divisions. Each peer belonged to a randomly chosen subnet. The propagation delay between an arbitrary peer and any of GTS, ORG, and SS was set at 200 msecs. Propagation delay between two arbitrary peers was set at 10 msecs independent of their locations in a branch. The fanout was three for all peers and ORG, but every peer could additionally have one direct peer. Only one video stream was available, which was coded at the constant rate of 1 Mbps and it was 186 seconds long. We employed PB with $\alpha = 2$. A video stream was divided into five segments whose first segment was six seconds long. Consequently, the length of the reception time was three seconds. We set the reserved time C at one second.

The number of new schedule requests per second called *arrival rate* was set at 30. Since the reception time lasts three seconds, the number of peers in a distribution

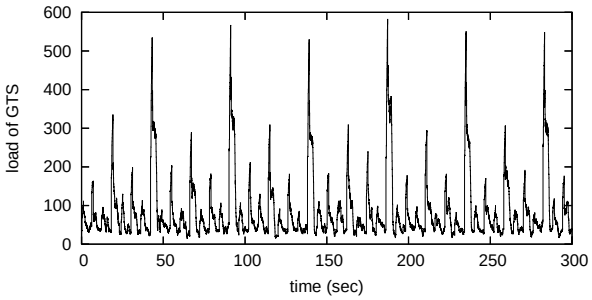

Figure 3. Transition of load of GTS

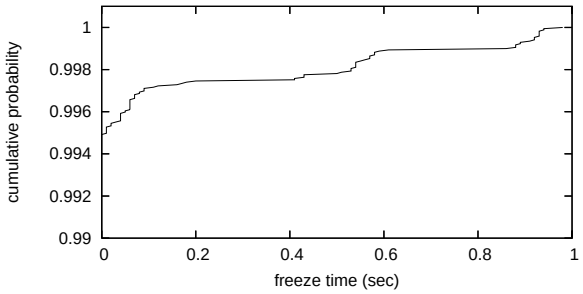

Figure 4. Distribution of freeze time

tree of the first segment amounts to 90 on average. Consequently, the number of peers participating in a tree of the second, third, fourth, and fifth segment becomes 180, 360, 720, and 1,440, respectively. It means that at an arbitrary instant, the number of peers participating in video distribution is 2,790 on average. We further assumed that every peer left the service without any notification at the probability of 0.004 at every second, even if it was on the way to participating in a tree, ready for the reception of a segment, receiving a segment, and trying recovering from a fault. Among 2,790 peers, about 11 peers left the service every second on average. Initial waiting time and freeze time of peers which left the service were not taken into account when we examined those measures.

We started each simulation experiment from a system with only ORG, GTS, and SS. We considered the instant when 144 seconds had passed since the first schedule request was sent as zero to eliminate the influence of initial conditions on evaluations. Every experiment lasted 1,440 seconds. In the following, values that averaged over ten simulation experiments are used.

3.2 Simulation results

Figure 3 illustrates the transition of the load of GTS. Since peers which were either STS or LTS in the preceding tree send participation requests to GTS at $t_i - W/2 - C$ for segment i on the assigned slot, we observe periodic spikes in the figure. The interval between the most highest spikes is 48 seconds, which corresponds to the instants when the beginnings of slots of all five segments are synchronized.

The highest value of the load of peer was 49, which corresponds to that of STS. On the other hand, an NP which participates a distribution tree as a leaf node does not receive any connection request. The average load was 4.8.

Since the inter-arrival time of two successive schedule requests follows uniform distribution, the resultant initial waiting time disperses almost uniformly. The average waiting time was 2.94 seconds. The maximum time of 5.35 seconds consists of a one-way propagation delay of 200 msecs from a peer to SS for schedule request, four seconds of $W/2 + C$, and the segment transfer delay of about 200 msec from ORG to the peer, where the peer is assumed to be LTS or STS. In addition, about one second is spent recovering from faults which accidentally occurred during the tree construction. The minimum time was 1.44 seconds.

Figure 4 depicts the distribution of fault recovery time. Among $1,440 \times 30 = 43,200$ peers joined the service, about 20,000 peers completed watching a whole video stream. The figure shows that about 100 peers, only 0.5% of 20,000 peers, experienced freezes due to the starvation of buffer. The maximum freeze time that one peer suffered was 0.98 seconds, short enough for users to tolerate.

We also conducted simulation experiments with a variety of parameter settings, although details are omitted due to space limitations. For example, with an arrival rate of 70, where 6,510 peers were participating in the service at an arbitrary instant, the load of GTS and peers increased in proportion to the rate. However, the average of fault recovery time decreased with the increase in the arrival rate because most faults occurred in inter-NP trees. It took only 20 msecs if a peer is accepted by its grandparent. When the probability of failures increased to 0.005, with which only 39% of 43,200 peers watched a whole stream, the maximum fault recovery time increased to 2.2 seconds. As a result, the maximum freeze time also increased, but was still smaller than one second.

4 Practical experiments

To verify the practicality of our scheme, we implemented our proposed algorithm and mechanism on an experimental environment and conducted several experiments. Due to space limitations, only some results will be shown.

We simulated a network with thousand NPs in a subnet by 19 hosts as depicted in Fig. 5. According to our tree construction mechanism, the depth of the tree becomes ten when the maximum fanout is two. For the last thousandth peer to join the tree, it takes $\frac{1,000}{417.12} + \frac{8}{417.12} = 2.42$ seconds on average except for the propagation delay, since requests were handled by STS one-by-one and the peer tried eight temporary parents. The value 417.12 is the average number of request that our peer can handle per second. On the other hand, the time required for a segment to traverse an

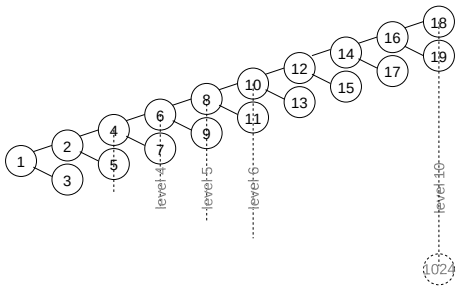

Figure 5. Distribution tree for practical experiments

NP tree and reach the tenth level ranged from 223 to 711 msecs in our experiments in a local area network. Those results mean that reserved time C should be set larger than 2.2 seconds so that the last peer has time to join a tree.

We also evaluated the initial waiting time. We randomly generated a thousand schedule requests and observed the time from when a play request was issued to the beginning of the video play-out. Depending on the instant that the service was requested by a user, the initial waiting time varies. However, a user could begin to watch a video in at most four seconds.

In the case of a single fault in an NP tree, the time from when a peer detects a fault in its parent to when it resumes receiving the stream from its grandparent ranged from 4 to 7 msecs. The average was 5.8, an acceptably small enough number. We considered the worst case scenario on an NP tree of 1,023 peers in which all the peers above nine-th level and STS die all at once. Detecting the fault of a parent, each peer on the tenth level first sends a connection request to its grandparent. After a while, it recognizes a fault of the grandparent, since it cannot communicate with the grandparent. Then it visits a great-grandparent. Those processes are repeated until a peer detects a fault in the STS. Eventually, all 512 peers on the tenth level send participation requests to GTS at almost the same time. The first peer is appointed the new STS and is introduced to the LTS. The rest of the peers are introduced to this new STS. Finally, peers construct a new distribution tree. The time required for the last peer to re-join the tree is 1,622 msecs except for propagation delays. It follows that a user does not notice the disaster as long as a peer deposits video data of more than about two seconds in its buffer.

5 Conclusion

In this paper, we proposed a hybrid scheme for video streaming on enterprise P2P networks where distribution trees are constructed considering the physical structure of underling networks. Our scheme can provide thousands or ten thousands of users with video streaming services in about five seconds and only 0.5% of users experience freezes less than one second.

In many other papers, they distribute video streams on flat P2P networks. For that reason, it is required to measure and estimate characteristics of network structures for constructing distribution trees considering underlying physical networks. However, it causes large overhead. Even if they can measure it with relatively light loads, they cannot estimate the structures instead, and their distribution trees are hardly regarded as to be adopted to physical networks. Our proposal can be adopted only to structured networks, which enables us to construct robust distribution trees easily and quickly. In addition, our proposal can use other better proposal for constructing inter-NP trees if each subnet has many NPs.

However, we also found an instantaneous increase in the load on the GTS. We need an improved mechanism to distribute the load efficiently while retaining the advantages of the distributed tree construction. In addition, some preliminary experiments show that redundant flow exists on inter-branch links. We propose a new algorithm to build bandwidth-efficient, low-cost inter-branch distribution trees.

References

[1] A. Nicolosi and S. Annapureddy, "P2PCAST: A Peer-to-Peer Multicast Scheme for Streaming Data," in *Proceedings of First IRIS Student Workshop*, Aug. 2003.

[2] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient Peer-to-Peer Streaming," in *Proceedings of IEEE ICNP 2003*, Nov. 2003.

[3] 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*, pp. 45–54, Nov. 2003.

[4] A. El-Sayed, V. Roca, and I. Rhone-Alpes, "A Survey of Proposals for an Alternative Group Communication Service," *IEEE Network Magazine special Issue on Multicasting*, Jan. 2003.

[5] S. Viswanathan and T. Imielinski, "Pyramid Broadcasting for Video on Demand Service," in *Proceedings of SPIE MCNC*, vol. 2417, pp. 66–67, Feb. 1995.

[6] K. A. Hua and S. Sheu, "Skyscraper Broadcasting: A new Broadcasting Scheme for Metropolitan Video-on-Demand System," in *Proceedings of SIGCOMM '97*, pp. 89–100, June 1997.

[7] L. Mathy, R. Canonico, and D. Hutchison, "An Overlay Tree Building Control Protocol," *Lecture Notes in Computer Science*, vol. 2233, pp. 78–87, 2001.

[8] B. Zhang, S. Jamin, and L. Zhang, "Host Multicast: A Framework for Delivering Multicast to End Users," in *Proceedings of INFOCOM 2002*, June 2002.