PAPER

Heterogeneous Video Multicast in an Active Network

Héctor AKAMINE[†], ***Student Member***, **Naoki WAKAMIYA**[†], ***and*** **Hideo MIYAHARA**[†], ***Members***

SUMMARY We present a simple framework for multicasting video in an active network, in which we overcome heterogeneity in the quality requests by filtering the video stream at some properly located active nodes. The framework includes the requirements for the underlying active network and outlines the video multicast application. We then introduce a heuristic algorithm for electing the filtering nodes to conform a multicast distribution tree, in which we use an objective function to, for example, minimize the required bandwidth. We evaluate the performance of our algorithm comparing it with simulcast and layered encoded transmission through simulation experiments, showing some advantages of using active filtering.

key words: *heterogeneous multicast, active network, video filtering*

1. Introduction

In a heterogeneous environment, clients joining the same multicast session can have different quality requests due to limitations in the network bandwidth or in the processing capabilities of the end hosts. Approaches for dealing with heterogeneity include simulcast and layered encoded video. In simulcast, the server produces a different video stream for each requested quality, thus wasting the network resources. In layered encoding [1], the video stream is decomposed into a base layer and some enhancement layers that provide limited granularity, with the drawback of the difficulty in generating and decoding layerd streams, and the bandwidth overhead.

Active networking is appealing because of the advantages that introducing programmability in the network can offer. It is expected that network protocols can be developed and deployed faster and easier than in the current Internet, and that novel protocols, such as new approaches for multicast [2, 3], can be introduced to the benefit of users.

Multicasting video to heterogeneous clients is an application that can benefit of the programmability offered by an active network. At video filtering nodes, new streams of lower qualities can be derived from the received ones, and hence we become able to satisfy diverse quality requirements. Active filtering seeks to reduce the use of the required bandwidth choosing the filtering nodes appropriately.

Research into filtering by Yeadon et al. [4] and Pasquale et al. [5] propose filtering propagation mechanisms to vary the location where filtering occurs according to the requirements of downstream clients. AMnet [6] proposes a model and an implementation for providing heterogeneous multicast services using active networking. According to this approach, a hierarchy of multicast groups is formed, in which some active nodes that act as receivers in a multicast group become roots in other multicast groups, but it is not explained how the multicast groups are conformed and how the root senders of each multicast group are elected.

In this work we aim at two objectives. First, we give a framework for heterogeneous video multicast, considering a network in which active nodes can perform filtering of the video stream to generate lower quality ones to satisfy requests of downstream clients. In our framework, we first collect all the session clients' requests, and form a hierarchy of multicast groups. The server of the original video stream becomes the root of the top level group. The members of this group are the clients which requested the highest quality video, and one or some active nodes which filter the video stream. These active nodes become roots of other multicast groups where the filtered stream is supplied to other clients. Analogously, these new multicast groups can have one or some active nodes as members that become roots of even lower level groups. Second, we propose and evaluate an algorithm to appropriately elect the roots of the multicast groups. The effectiveness of active filtering depends on the topology of the video distribution tree, but to our knowledge no previous work has discussed this issue. In the proposed algorithm, we find a multicast tree that has the lowest value of an objective function that considers the network resources, such as the required bandwidth.

Details on the implementation of filtering mechanisms at active nodes are out of the scope of this work. Work in filtering implementations can be found in [4, 7, 8].

The rest of this paper is organized as follows: Section 2 describes our framework for video multicast using active filtering; Section 3 gives the details of the algorithm for electing an appropriate multicast distribution tree; Section 4 evaluates its performance, comparing it with other approaches for distributing video; Section 5 concludes our work.

2. A Framework for Heterogeneous Video Multicast Applications

2.1 Heterogeneous Video Multicast Applications

We call *heterogeneous video multicast* to a multicast session in which all its members receive the same video sequence, but not necessarily with the same quality. We consider a

Manuscript received January 17, 2001.
Manuscript revised July 23, 2001.

[†]The authors are with the Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University

video server, which whenever possible sends a unique video stream, and a set of clients which can elect from a discrete set of available quality levels. The network consists of conventional and active nodes, in which the latter can perform filtering when required, to satisfy different quality requirements.

2.2 Characteristics of the Active Network

We assume a network in which some of the nodes are active. According to the framework proposed by the Active Network Working Group [9], an active node can handle both conventional and active packets, which are distinguished by the presence of an *Active Network Encapsulation Protocol* (ANEP) header. This framework presents a structure for active nodes with three major components: the *Node Operating System* (NodeOS), which manages the node resources such as link bandwidth, CPU cycles and storage; the *Execution Environments* (EEs), each one of which implements a virtual machine that interprets active packets that arrive at the node; and the *Active Applications* (AAs), which program the virtual machine provided by an EE to provide an end-to-end service. End systems that host end-user applications are also considered as active nodes having the same structure.

Active nodes differ from conventional nodes in that they have memory and processing resources. Using those resources, end users can customize the network behavior. Limitations on programmability of active nodes are indeed imposed by the different implementations [10, 11]. For supporting the proposed approach, the underlying active network implementation must provide programmability in these two aspects:

1. *Protocol signaling*, the video application needs means for leaving soft-state in or getting information from active nodes. The required processing overhead is low.
2. *Filtering code*, the program to be executed on the active node when the incoming video data needs transformation to reduce its size. In general the size of the filtering code is relatively large, resource consuming, and requires fast execution.

Some existing implementations divide network programmability in a way that can fit with the above requirements. SwitchWare [12] divides code between *active packets* and *active extensions*. Active packets replace traditional packets and carry both data and code with limited functionality, and they are used for inter-node communication. Active extensions can be dynamically loaded to give nodes added functionality. ANTS [13] has a code propagation mechanism to load program code into a node when an arriving capsule requires execution of code unavailable at the node. It also implements a mechanism called *extensions* to allow code with privileges or whose size is too large to be transferred using its capsule-based code distribution mechanism. AMnet [6] uses out of band signaling to load *modules* into the nodes that require code to transform the in-band data.

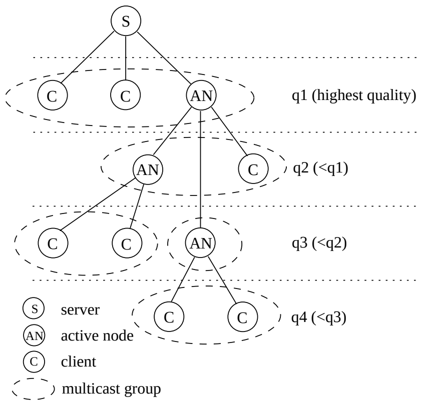

Fig. 1 Example of a logical topology.

Considering the complexity and the size of the filtering code, it would be better to have the code preloaded, using a node set up phase that we describe in the next subsection, before the beginning of the multicast transmission. This set up phase can be avoided and use on-the-fly mechanisms, such as ANTS' code propagation, if the code doesn't involve such complexity and size.

The application should have information on the network to construct an appropriate video distribution tree. Since active nodes have the functionality of conventional nodes and support non-active traffic, we assume that standard network layer protocols such as OSPF [14] can be used to discover the network topology. Similarly, it is necessary to consider a signaling protocol between active nodes in order to exchange information exclusive to them. Due to security concerns, the information that active nodes exchange must be limited. We assume that such kind of protocol already exists and an active node can "discover" other active nodes in the network and query some basic information such as the EEs running on them.

2.3 Outline of the Application

Our approach for heterogeneous video multicast considers filtering at some properly located active nodes. The video server sends the video stream of the highest quality among all the clients' requests. To satisfy lower quality requests, some active nodes are designated as filters. A designated active node first subscribes to the corresponding multicast group to receive the stream to transform, then it filters/transcodes the stream, and becomes root of a new multicast group of which the clients requesting the transformed video stream are members. We can re-filter an already filtered video stream in order to obtain another one with lower quality, and hence a hierarchy of multicast groups can be conceived. This idea is used in AMnet [6]. Figure 1 depicts this approach.

Each multicast group in the hierarchy is constituted using network layer multicast (i.e., IP multicast). Those groups are "glued" and ordered hierarchically using a protocol implemented for the active network.

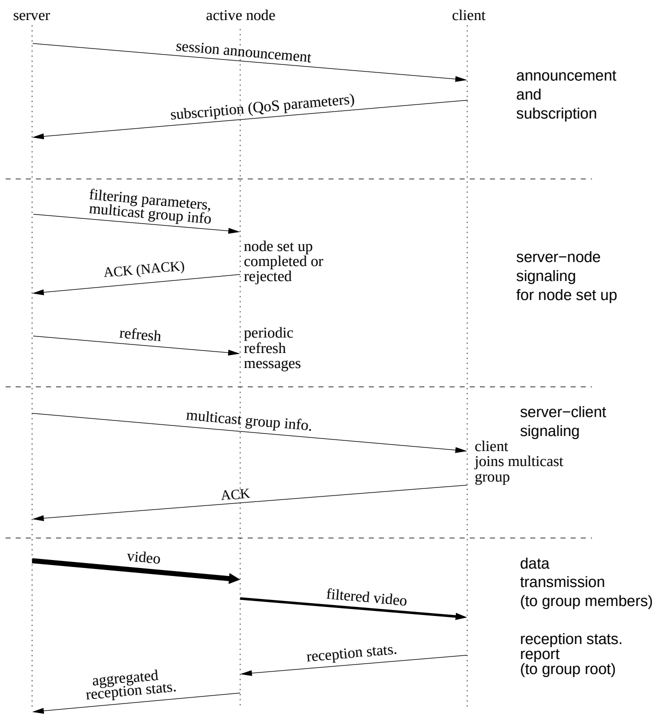
server
active node
client
session announcement
announcement and subscription
subscription (QoS parameters)
filtering parameters, multicast group info
node set up completed or rejected
ACK (NACK)
server−node signaling for node set up
refresh
periodic refresh messages
multicast group info.
server−client signaling
client joins multicast group
ACK
video
filtered video
data transmission (to group members)
reception stats.
reception stats. report (to group root)
aggregated reception stats.

Fig. 2 Signaling for the active application.

We now describe the components for an application that uses filtering to multicast video to satisfy heterogeneous requests. We schematized it in Figure 2.

1. *Session announcement*. The server uses a well-known multicast address to inform the possible clients about the session, including information such as the available quality levels and the required amount of resources. The protocol used can be similar to the SAP protocol [15] used in the MBONE.
2. *Session subscription*. Using the information from the session announcement, each of the clients that wants to participate in the session sends a request to the server containing the desired quality parameters. The quality requested by the client reflects the user's preference on the perceived video quality and limitations on its available resources [16]. In the algorithm in Section 3, we assume that quality corresponds to one QoS dimension for simplicity, but it is possible to consider more parameters, e.g., quantization scale and frame rate.
3. *Derivation of the distribution tree*. After the requests are collected, the server defines the conformation of the multicast groups and the active nodes that are going to perform filtering considering the network condition, i.e., topology, resource availability and clients' requests. The calculation algorithm is explained and evaluated in the following sections.
4. *Set up of filtering nodes*. We assume filtering code to be preloaded prior to sending the video stream. We require a signaling procedure to inform the designated nodes that they are roots of multicast groups and to load the required filtering program. In the case that the designated node cannot afford to filter due for example to insufficient resources, we go back to the previous step and choose a new node and therefore a different distribution tree.
   For node set up, the server must send the following information:
   - Multicast address as receiver: the active node receives the video data to be filtered as a member of this multicast group.
   - Multicast address as sender: the active node distributes the filtered video data using this multicast address.
   - Filtering parameters: the sender sends a reference to the required code, and the required parameters, e.g., the quantization scale in a requantization filter. As explained before, a designated filtering node must pre-load the filtering program using the reference. If it is not possible, set up fails.

   The session is maintained in soft state. After set up, "refresh" messages are periodically sent in order not to allow the node to release the reserved resources assuming that the filtering is no longer needed.
5. *Client subscription to the multicast group*. We are assuming to use the existing IP multicast protocols, such as IGMP for client-router communication, and DVMRP and MOSPF between routers [17]. We assume the server and the session clients to be in the same routing domain, for reasons given in the next section. IP multicast requires each client to join a multicast group specifying the group IP address. In our approach, the sender informs each client of the IP addresses of the multicast groups which it should subscribe. On receiving the multicast group address, the client performs the corresponding subscription.
6. *Data transmission and feedback*. The server multicasts the video stream of the highest quality to requesting clients and active nodes which filter it to get the lower quality streams.
   Although not discussed in this work, it is necessary to monitor the reception conditions of the clients, since available bandwidth for the video session is not assured in best effort networks. The use of active nodes and a hierarchy of groups can help to control feedback implosion. Each client sends feedback messages only to the root of the multicast group to which it is subscribed. Results are consolidated by the active node acting as root, which in turn sends a report containing its own reception condition information and/or consolidated information of its multicast group to the root of the parent multicast group.
   Parameters of interest for video applications include packet loss, delay and jitter. We can use RTP packets [18] to send the video data, and then infer those parameters. The use of this information to dynamically modify the distribution tree is left for future study.

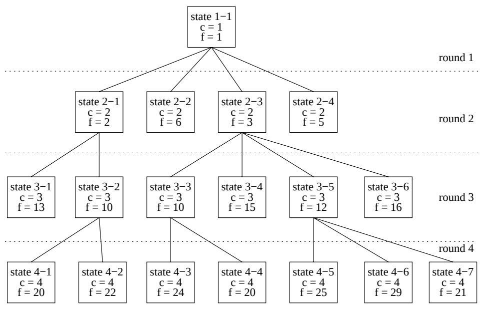

Fig. 3 Example of a state tree.

3. Algorithm for Construction of the Multicast Distribution Tree

As described in Section 2, the multicast distribution tree consists of a hierarchical conformation of multicast groups, and the purpose of the algorithm is to adequately elect the root and members of each of them and to choose on which active nodes filtering must be done.

For simplicity, we assume one QoS dimension, and a scalar value denotes the requested quality. We also assume that the paths the multicast routing algorithm uses are the same as the unicast paths from the source to each one of the destinations, as created by Dijkstra's algorithm, since this coincides with multicast routing algorithm used in dense mode subnetworks such as DVMRP and MOSPF. Since the algorithm uses a centralized approach in which the network topology and the set of requests are known by the video server, its applicability would be limited to sessions with all the members in the same domain due to scalability concerns. Thus, all our assumptions consider an intra-domain case and do not cover issues such as inter-domain multicast.

3.1 Algorithm Description

The algorithm forms a distribution tree in a request by request basis, taking the requests in descending order of quality, and in ascending order of distance from the sender in the case that requests of the same quality exist.

Each step in the construction of the tree defines a state. A state is defined by:

1. The number of clients' requests c that have been already considered in the tree.
2. The characteristics of the distribution tree needed to serve those requests, that is, used links and filtering nodes
3. The value of an objective function f, which is a measure of how good is the distribution tree being formed. A lower value of f means a better distribution tree. The definition of f can consider factors such as the total used bandwidth, link utilization and/or the use of node resources.

Figure 3 depicts a sample state tree. Each state is denoted as $c - i$, where c stands for the number of clients, $i \leq N_c$ is the state index, and N_c is the number of states in that round. At the first round, there is only one state 1-1, where only one client with the highest demand is satisfied by being provided the video stream at the required quality directly from the server. From a state in round c, it is possible to derive several states for round $c+1$, depending on how the stream that the new client demands has been generated.

When we consider a client to be added to the distribution tree in the next round, we define a set of "candidate senders" for that client. Either the original server of the video sequence or any of the active nodes in the network can be the candidate sender. Each state in the round corresponds to one of the candidate senders. For a given flow request and candidate sender, one of the following conditions holds:

1. The candidate sender is already requested to relay a stream with the desired quality by a previously processed client. In this case the client subscribes to the multicast group the stream belongs to.
2. The candidate sender is already requested to relay a stream with a quality higher than the one requested. In this case, this stream must be filtered at this candidate sender. Then, a new multicast group is created with the candidate sender as the root, and the requesting client becomes a member of this multicast group.
3. The candidate sender is not relaying a flow. In this case, the candidate sender must first subscribe to a multicast group, filter the stream that receives as a member of this group, and become the root of a new multicast group. The requesting client subscribes to this new group to get the stream.

For simplicity, we assume only one variable that comprises the node resources, and that a filtering operation reduces the value of this variable by a predefined amount. If one active node has already exhausted its resources, filtering cannot be performed, and it is not considered as a candidate sender.

Doing an exhaustive search, considering all the possible combinations, can lead to the generation of a very large number of states. In the worst case, the number of candidate senders is equal to the number of active nodes in the network, say A, plus the original server. In such a case, the number of states N_c in round c becomes $(A+1)^{c-1}$. Since this is computationally expensive if the number of requests or active nodes in the network is not small, two parameters were defined to restrict the number of states N_c to analyze:

- We limit the number of candidate senders to expand in each round to a fraction b of the total candidate senders.
- We restrict the number of states to expand in a round to a maximum of m.

In each round, we select up to a maximum of m states to expand, the states chosen are the ones with the lowest values of f. Each state is expanded with $b \times (A+1)$ new states, in which each new state implies a different candidate sender elected to satisfy the request of the next client. The election of these new states is done by the distance in num-

Table 1 Required bandwidth for streaming video (Mb/s).

quality (quantizer scale)	single-layer video	layered video
4 (10)	14.4	22.65
3 (20)	8.8	13.64
2 (30)	6.6	8.75
1 (40)	5.4	5.19

ber of hops, between the client and the candidate node: the first candidate to choose is the closest node to the client that already belongs to the distribution tree, i.e., that relays or filters a flow to satisfy requests of previous rounds. The next candidates are chosen close to this one.

In this paper, we have not analyzed the effect of the values of b and m, and we chose them empirically for our evaluation experiments. We continue expanding the state tree until all the clients' requests are satisfied. Then, the state with the lowest f is chosen.

3.2 Example

Figure 4 shows an example network with 10 nodes. Active nodes are marked with squares and non-active ones with circles. Client requests are indicated with unfilled circles with a number that represents the requested quality. The server is attached to node 3. When the sender is attached to an active node, we must distinguish if the filtering is performed at the active node, or if the stream is provided by the sender.

The qualities are related with the bandwidth according to the data in Table 1, taken from a previous work from our research group [19] for the MPEG-2 video coding algorithm [20]. In layered video, the layers must be piled up to achieve higher quality video. For example, the bandwidth required for a stream of quality 4 is given as 5.19 (layer 1) + 3.56 (layer 2) + 4.89 (layer 3) + 9.01 (layer 4) = 22.65 Mb/s. The different qualities are obtained varying the quantizer scale, and active nodes derive the video stream of lower quality by de-quantizing and re-quantizing the received stream.

Figure 4 shows the multicast groups (mg1, mg2, ... mg5) conformed by our algorithm. Arrows show the required streams, and arrow tips point to multicast group members. Two filtering processes are needed in node 4 and one in node 9. It must be noted that active node 4 becomes member of multicast group mg1, just to provide filtered streams to clients in nodes 1 and 6.

4. Evaluation

In this section, we show the effectiveness of the proposed algorithm through some numerical experiments. We generate random topologies using Waxman's algorithm [21], and choose the parameters appropriately to generate topologies with an average degree of 3.5, to try to resemble the characteristics of real networks [22]. We assumed the proportion of active nodes in the network to be 0.5. For simplicity, each filtering operation is assumed to use the same amount of resources. We also assumed that the number of filtering

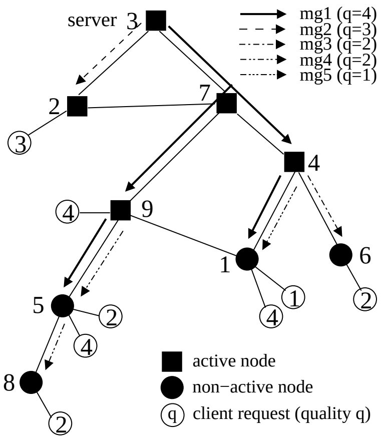

Fig. 4 Multicast groups for our algorithm.

operations that each active node can do is a random value between 15 and 30. The location of active nodes is chosen at random. The location of the server, the clients and their corresponding requests' qualities are also generated randomly, and vary from one experiment to the other. Clients can request the video stream in one of four available video qualities, according to Table 1.

For comparison, we also used approaches based on simulcast and layered coded video to construct the multicast trees, using the same topologies and requests data.

As we state in the previous section, the definition of f can be modified according to which network parameters are most important in the construction of the distribution tree. We performed the evaluation using two simple definitions as detailed below.

4.1 Comparison of Required Bandwidth

First, we compare the total used bandwidth for the multicast distribution tree. We define f_1 as

$$f_1 = \sum_{i \in \mathcal{U}} B_i \qquad (1)$$

where i denotes a link, $\mathcal{U}$ is the set of used links in the distribution tree described by a state, and B_i denotes the bandwidth devoted to video distribution in link i.

To evaluate our algorithm with f_1, we generate ten 20-node and ten 50-node network topologies, varying the number of requests among 10, 20 and 50. The results are summarized in Figs. 5 and 6. In both figures, the first 10 experiments are for sessions composed of 10 requests (labeled as “10r”), the next 10 for 20 requests (“20r”) and the last 10 for 50 requests (“50r”).

In general, the proposed algorithm requires less bandwidth than simulcast and layered video, at the cost of processing power of the filtering nodes. When the number of requests is small, the total bandwidth used by simulcast transmission is even smaller than the one required for layered transmission, because the overhead of the latter is not justified owing to the dispersed clients. As the number of re-

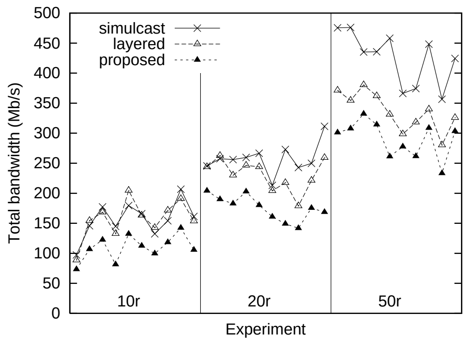

Fig. 5 Total bandwidth, 20-node network.

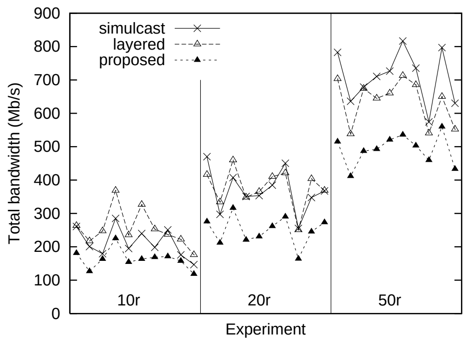

Fig. 6 Total bandwidth, 50-node network.

quests increases, the bandwidth required for layered encoding transmission becomes less than the required by simulcast, and becomes closer to the one required by our proposed algorithm. Since we fixed the proportion of active nodes to be 0.5 in the generated topologies, when we increase the number of requests, the number of streams relayed from filtering nodes to the non-active ones (to satisfy requests from clients attached to the non-active nodes) can also increase. Then, the required bandwidth of the multicast tree becomes larger, reducing the advantage of the proposed algorithm.

4.2 Simultaneous Multicast Sessions

Minimization of the total bandwidth required for video multicast is intended to avoid the extremely high load on the network and leave resources to other users. In this subsection, we compare our algorithm to see how many sessions can be simultaneously set up.

In the experiments, all the links are assumed to have a bandwidth of 100 Mb/s. Sessions are set up until the bandwidth of any link is exhausted. Here, we should note that the network we consider is best-effort and the constraint on the available link bandwidth is not directly taken into account in the proposed algorithm stated in Section 3. Thus, the results can be regarded as the number of simultaneously acceptable sessions without causing a seriously overloaded link. The sessions are independent, and we do not use the information of the links used by the other sessions to build the current tree.

We introduce another definition for the objective function, f_2, which is related to the required average bandwidth of the used links:

$$f_2 = \frac{\sum_{i \in \mathcal{U}} B_i}{|\mathcal{U}|} \qquad (2)$$

With this definition, the algorithm tries to reduce the average link bandwidth required by the session, spreading the multicast tree in more links. We expected that with the use of more links per tree we could avoid link congestion and therefore increase the number of simultaneous sessions.

In Figs. 7 and 8, we labeled the experiments 20n10r, 20n20r, 20n50r, 50n10r, 50n20r, and 50n50r, where the first number denotes the nodes in the network, and the second number the requests per session.

Figure 7 shows the average bandwidth required to establish the first ten sessions at the same time. f_1 shows the lowest value for all the cases. Even though we chose f_2 to minimize the average bandwidth consumption on the links, when we sum all the sessions, f_2 results in the highest values. Between them lie the values for simulcast and layered video. When the number of requests is small (10 requests), the average bandwidth used by layered encoded distribution is greater, but for larger number of requests it is surpassed by the values of simulcast.

Figure 8 shows the maximum number of simultaneous sessions that could be set up. The results show performance in the following order, from better to worse: the proposed algorithm using f_1, the proposed algorithm using f_2, layered transmission and simulcast. There were few cases in which our proposed algorithm was surpassed by the layered video approach. This occurs, for example, when we have several clients connected to a non-active node that request different quality streams.

Even when the location of senders are concentrated in a region of the network, the advantage of f_2 is relatively small although results are not shown in this paper. With f_2 we expected to increase the number of possible simultaneous sessions, reducing the bandwidth consumption per link, at the expense of increasing the number of used links. However, this increase was done greedily and did not improve the results.

4.3 Computation Time

We assumed that simulcast, layered multicast and the proposed approach create source-rooted multicast groups conformed by the unicast paths from the source to the clients, using Dijkstra's algorithm. In simulcast and layered multicast, there is only one possible solution, since trees are constructed per requested quality and layer, respectively. In the proposed approach, different possible partial multicast trees composed of a hierarchy of multicast groups are tried to find a solution that reduces the value of the objective function. Then, we measure the required time by the algorithm for electing the composition of the multicast groups rather than the routing algorithm.

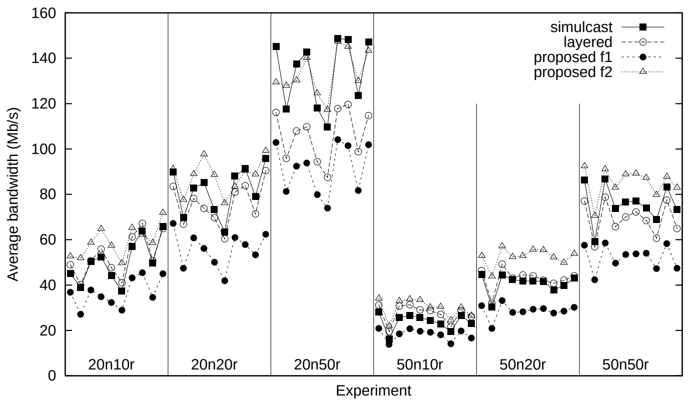

Fig. 7 Average bandwidth (Mb/s) for the first ten sessions.

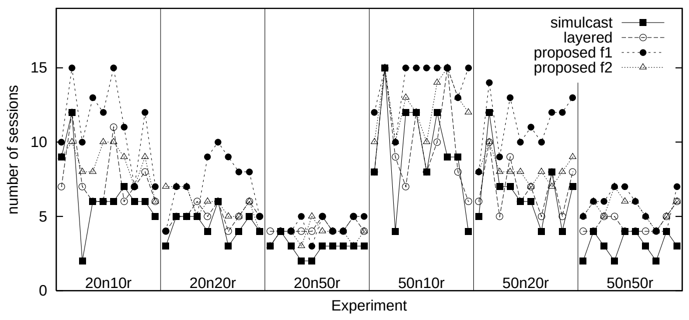

Fig. 8 Maximum number of simultaneous sessions (up to 15).

We evaluated a Java implementation of the algorithm on an 800 MHz Pentium III machine. We have not analyzed the effect of varying the values of the parameters b and m. Their election involves a trade-off between required processing time and optimality of the obtained solution. Just to have an insight, we averaged the time required by our algorithm to generate the multicast trees using empirical values for b and m. For networks with 20 nodes, we required an average of 6, 14 and 56 seconds for trees with 10, 20 and 50 requests, respectively, using $m = 20$ and $b = 1$. For networks with 50 nodes, we required 101, 238, and 451 seconds for 10, 20 and 50 requests, respectively, using $m = 20$ and $b = 0.3$ for the first two cases and $b = 0.2$ for the last case. We can interpret these values as the algorithm having low scalability for higher numbers of nodes and clients.

5. Conclusions and Future Work

We outlined a framework for multicasting video to a heterogeneous group of clients, considering a network in which active nodes can perform filtering of the original video stream to satisfy different quality requirements. We then presented an algorithm for electing the filtering nodes in this distribution tree, which uses a function f that can be set to consider the efficient use of network resources. We evaluated our algorithm choosing two simple definitions for f: the total bandwidth used, i.e., the sum of the bandwidth used in each link, and the average bandwidth of used links, compared it with simulcast and layered multicast, and found that using our algorithm we can achieve a more effective use of the available bandwidth of the network, but at the expense of requiring processing capability at the network nodes. We assumed a best-effort network, but it is possible to apply our algorithm for reservation based networks. In that case, we can consider other variables in the tree construction such as link capacity, available bandwidth, or utilization.

We evaluated the performance of the algorithm considering a random location of requests. Due to the reason that higher quality requests are processed first and the way candidate senders are elected to conform the hierarchy of multicast groups, we think that the algorithm is more effective in the case quality requests show some correlation between their locations and with their distances to the server. Problems with the algorithm include its low scalability, due to the use of a centralized approach that requires knowledge of the entire topology and the set of requests by the server, and its inability to modify the tree dynamically.

Future research topics include the consideration of the effect of delays introduced at the filtering nodes, interdomain issues in bigger topologies, and the use of reception feedback to dynamically modify the multicast tree after the beginning of the video transmission.

Acknowledgments

This work was partly supported by Research for the Future Program of Japan Society for the Promotion of Science under the Project "Integrated Network Architecture for Advanced Multimedia Application Systems," a Grant-in-Aid for Encouragement of Young Scientists 12750322 and Special Coordination Funds for Promoting Science and Technology from The Ministry of Education, Science, Sports, Culture and Technology of Japan, and the Telecommunication Advancement Organization of Japan under the Project "Global Experimental Networks for Information Society."

References

[1] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven Layered Multicast," Proc. ACM Sigcomm, pp. 117-130, August 1996.

[2] M. Calderón M. Sedano, A. Azcorra, and C. Alonso, "Active Network Support for Multicast Applications," IEEE Network, pp. 46-52, May/June 1998.

[3] L. Lehman, S. Garland, and D. Tennenhouse, "Active Reliable Multicast," Proc. IEEE Infocom, pp. 581-589, 1998.

[4] N. Yeadon, F. García, D. Hutchinson, and D. Shepherd, "Filters: QoS Support Mechanisms for Multipeer Communications," IEEE Journal on Selected Areas in Communications, Vol. 14, No. 7, pp. 1245-1262, September 1996.

[5] J. Pasquale, G. Polyzos, E. Anderson, and V. Kompella, "Filter Propagation in Dissemination Trees: Trading Off Bandwidth and Processing in Continuous Media Networks," Proc. NOSSDAV, November 1993.

[6] B. Metzler, T. Harbaum, R. Wittmann, and M. Zitterbart, "AMnet: Heterogeneous Multicast Services based on Active Networking," Proc. IEEE OpenArch, pp. 98-105, March 1999.

[7] M. Hemy, U. Hengartner, P. Steenkiste, and T. Gross, "MPEG System Streams in Best-Effort Networks," Proc. Packet Video, April 1999.

[8] S. Gopalakrishnan, D. Reininger, M. Ott, "Realtime MPEG System Stream Transcoder for Heterogeneous Networks," Proc. Packet Video, April 1999.

[9] K. Calvert, ed., "Architectural Framework for Active Networks, Version 1.0," Active Network Working Group Draft, 1999.

[10] D. Tennenhouse, J. Smith, D. Sincoskie, D. Wetherall, and G. Minden, "A Survey of Active Network Research," IEEE Communications Magazine, pp. 80-86, January 1997.

[11] J. Smith, K. Calvert, S. Murphy, H. Orman, and L. Peterson, "Activating Networks: A Progress Report," IEEE Computer, pp. 32-41, April 1999.

[12] S. Alexander, W. Arbaugh, M. Hicks, P. Kakkar, A. Keromytis, J. Moore, C. Gunter, S. Nettles, and J. Smith, "The Switchware Active Network Architecture," IEEE Network, Vol. 12, No. 3, pp. 27-36, May 1998.

[13] D. Wetherall, "Service Introduction in an Active Network," Ph.D. Thesis, Massachusetts Institute of Technology, 1999.

[14] A. Tanenbaum, "Computer Networks," 3rd. edn., Prentice Hall, 1996.

[15] M. Handley, "SAP: Session Announcement Protocol," Internet Draft, Work in Progress, 1996.

[16] K. Fukuda, N. Wakamiya, M. Murata, and H. Miyahara, "On Flow Aggregation for Multicast Video Transport," Proc. Sixth IFIP International Workshop on Quality of Service, pp. 13-22, May 1998.

[17] C. Semeria, and T. Maufer, "Introduction to IP Multicast Routing," 3COM White Paper, available at `http://www.3com.com`.

[18] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," Request for Comments 1889, 1996.

[19] K. Fukuda, N. Wakamiya, M. Murata, and H. Miyahara, "Real-Time Video Distribution with Hybrid Hierarchical Video Coding in Heterogeneous Network and Client Environments, Proc. MMNS, November 1998.

[20] ISO/IEC DIS 13818-2: MPEG-2 Video. ISO Standard, 1994.

[21] B. Waxman, "Routing of Multipoint Connections," IEEE Journal on Selected Areas in Communications, Vol. 6, No. 9, pp. 1617-1622, December 1988.

[22] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," Proc. IEEE Infocom, March 1996.

Héctor Akamine received the B.S. degree in Electronic Engineering and the title of Electronic Engineer from the Pontifical Catholic University of Peru in 1996 and 1997, respectively, and the M.E. degree in Computer Science from Osaka University in 2001. He is currently a student at the Graduate School of Engineering Science, Osaka University. His research interests lie in the areas of multimedia communications, video multicast, and active networking. He is a student member of IEICE and IEEE.

Naoki Wakamiya received the M.E. and Ph.D. degrees from Osaka University in 1994 and 1996, respectively. He was a Research Associate of the Graduate School of Engineering Science, Osaka University, from April 1996 to March 1997, and a Research Associate of the Educational Center for Information Processing, Osaka University, from April 1997 to March 1999. He is an Assistant Professor of the Graduate School of Engineering Science, Osaka University, since April 1999. His research interests include performance evaluation of computer communication networks, and distributed multimedia systems. He is a member of IEICE, ACM and IEEE.

Hideo Miyahara received the M.E. and D.E. degrees from Osaka University in 1969 and 1973, respectively. From 1973 to 1980, he was an Assistant Professor in the Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University. From 1980 to 1986, he was an Associate Professor in the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. From 1986 to 1989, he was a Professor of the Computation Center, Osaka University. Since 1989, he has been a Professor in the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. From 1995 to 1998, he was the Director of the Computation Center of Osaka University. From 1998 to 2000, he was the Dean of the Faculty of Engineering Science, Osaka University. Since 2000, he is the Director of the International Student Center of Osaka University. From 1983 to 1984, he was a Visiting Scientist at IBM Thomas J. Watson Research Center. His research interests include performance evaluation of computer communication networks, broadband ISDN, and multimedia systems. Prof. Miyahara is a member of the IPSJ and an IEEE fellow.