An Approach for Heterogeneous Video Multicast Using Active Networking

Héctor Akamine, Naoki Wakamiya, Masayuki Murata, and Hideo Miyahara

Department of Informatics and Mathematical Science
Graduate School of Engineering Science, Osaka University
1–3 Machikaneyama, Toyonaka, Osaka 560–8531, Japan
Tel: +81–6–6850–6588 Fax: +81–6–6850–6589
akamine@ics.es.osaka-u.ac.jp

Abstract. We present a framework for heterogeneous video multicasting, considering an active network in which active nodes can filter the video stream to satisfy different quality requests. As a part of this approach, we propose a heuristic algorithm for the construction of a multicast distribution tree that appropriately chooses the active nodes at which filtering is performed with the aim of, for example, minimizing the total required bandwidth. We evaluate the performance of our algorithm and compare it against two other approaches: simulcast and layered encoded transmission. Through simulation experiments, we show that a larger number of simultaneous multicast sessions can be set up with active filtering.

Keywords: heterogeneous multicast, active networking, video filtering

1 Introduction

Heterogeneous multicasting of video is a natural candidate for enjoying the benefits of active networking. 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. Alternatives to dealing with heterogeneous quality requests for video multicasting include simulcast and distribution of layered encoded video [1], that have the advantage of being able to be used in the actual network infrastructure, but with excessive use of network resources. Active filtering seeks to reduce the use of the required bandwidth choosing the location of filtering nodes appropriately, with the compromise of requiring processing overhead at some nodes.

Research into filtering by Yeadon et al. [2] and Pasquale et al. [3] predates active networking research, but propose a filtering propagation mechanism to vary the location where filtering occurs according to the requirements of downstream clients. AMnet [4] 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 multicasting, considering a network in which active nodes can perform filtering

of the video stream to generate lower quality ones to satisfy requests of downstream nodes. In our framework, we first collect all the session clients' requests, and use this information to form a hierarchy of multicast groups, where the top level group root is the video server. 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, producing one with lower quality. These active nodes become roots of other multicast groups to satisfy the requirements of 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 efficiently 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.

The rest of this paper is organized as follows: Section 2 describes our framework for multicasting video using active node 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 Multicasting Applications

2.1 Assumptions about the Network

We assume a network in which some of the nodes are active. A proposed framework for active networks [5] presents a structure for active nodes, which is divided into three major components: the *Node Operating System* (NodeOS), which allocates 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. This framework also defines an *Active Network Encapsulation Protocol* (ANEP) header, which must be included in each active packet to distinguish to which EE it must be sent to be processed. Legacy (non-active) traffic whose packets don't include the ANEP header must also be supported by active nodes, and in this case they act as conventional nodes.

Active nodes differ from conventional nodes in that they have memory and processing resources that can be used by end users to customize the network behavior. We assume that AAs can leave state in the active nodes when necessary, i.e., put values or identifiers that will be used by subsequent packets of the same application.

Code that runs in an active node can be classified into trusted and untrusted. We call *trusted code* to the programs whose whose execution is previously known to be safe, i.e., will not harm the node. We can consider them as modules that enhance the node functionality, with few restrictions in the use of node resources. On the other hand, the execution of *untrusted code* must be enforced to remain between some limits (i.e., restricting the API it can use, limiting its size, the time and memory resources it can

consume.) It is usually executed through an interpreter to enforce security checking, slowing down its execution.

Some existing implementations make this distinction. SwitchWare [6] divides code between *active packets* and *active extensions*, active packets replace traditional packets and contain data and code but with limited functionality, and they are used for inter-node communication, while active extensions can be dynamically loaded to give nodes added functionality. ANTS [7] uses *extensions* to allow the existence of code with privileges or whose size is too large to be transferred using its capsule-based code distribution mechanism.

Video filtering code, excepting the simplest filter types, is relatively large, resource consuming, and requires fast execution, and is not likely to fit into the limitations of untrusted code. Handling the filter programs as trusted modules, we can use the advantage that video streams have a relatively large number of packets that require the same type of processing, by preloading the filtering code to work over the entire stream. Handling the code as trusted can also let us optimize it for faster execution.

We assume that the active network has mechanisms for loading trusted code into the nodes (e.g., trusted code servers from which it can be transferred.) We don't discuss security issues in this paper.

In contrast, code used for protocol signaling is used mainly for leaving soft-state in or getting information from active nodes, and can fulfill its purpose running within the restrictions of untrusted code.

To construct the video distribution tree appropriately, the algorithm should have information on the network. Since active nodes have the functionality of conventional nodes and support non-active traffic, we assume that they communicate with conventional nodes using standard network layer protocols such as OSPF [8], in order to discover the network topology. Similarly, it could be 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 using this must be limited. We assume that such kind of protocol already exists and it lets an active node to "discover" other active nodes in the network and query some basic information such as the EEs running on them.

2.2 Sketch of the Application

Our approach for heterogeneous video multicasting considers filtering at some properly located active nodes. The video server is required to produce and send the video stream of the highest quality among all the clients' requests. Then it is transcoded to a lower quality at one or more intermediate active nodes to, for example, fit to the available bandwidth of downstream links. Since the distribution of the video streams with different qualities is done using current multicast protocols, when the video stream needs such transformation, the 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

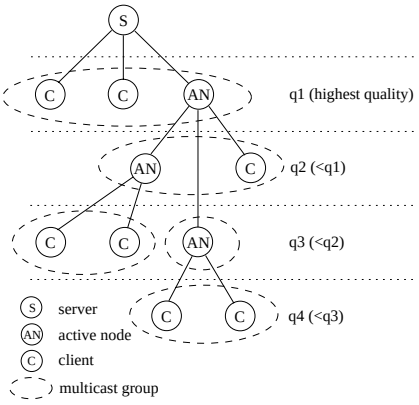

Fig. 1. Example of a logical topology

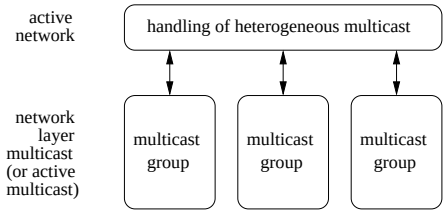

Fig. 2. Realization of heterogeneous multicast

groups can be conceived. This idea was pioneered in AMnet [4]. Fig. 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. It can also be possible to have active multicast protocols that replace network layer multicast in some groups, but similarly the interaction between multicast groups is controlled by an upper layer active protocol. This is shown in Fig. 2.

Yeadon et al. [2] presented some different approaches for filtering MPEG video streams. The simplest method for rate reduction is mere picture discarding, which consists on eliminating progressively B, P and I pictures. This approach is of limited applicability, since it only allows the reduction of bandwidth modifying the frame rate. Beyond picture discarding, other approaches include partial decoding and re-encoding of the video streams. For example, low-pass filters involve discarding the high frequency DCT coefficients, and requantization filters increase the value of the quantization factor to increase the number of zero DCT coefficients. They implemented those filters for MPEG-1 video streams, and found that although end-to-end delay and jitter is increased, they are feasible for continuous media streams. Here we do not specify which filtering approach to use, and assume that any one is usable. Nevertheless, it is necessary to consider that depending on the complexity, some filters could not be possible to implement or could introduce non-negligible delays at the active nodes.

We now describe the components for an application that performs heterogeneous video multicast employing filtering. We schematized it in Fig. 3.

1. *Session announcement*. The server uses a well-known multicast address to inform the possible clients about the session contents. Information includes the available qualities and required amount of resources. The protocol used to send these messages can be similar to the SAP protocol [9] used in the MBONE.
2. *Session subscription*. Each of the clients that wants to participate in the session sends a request to the server containing the desired quality parameters. The quality

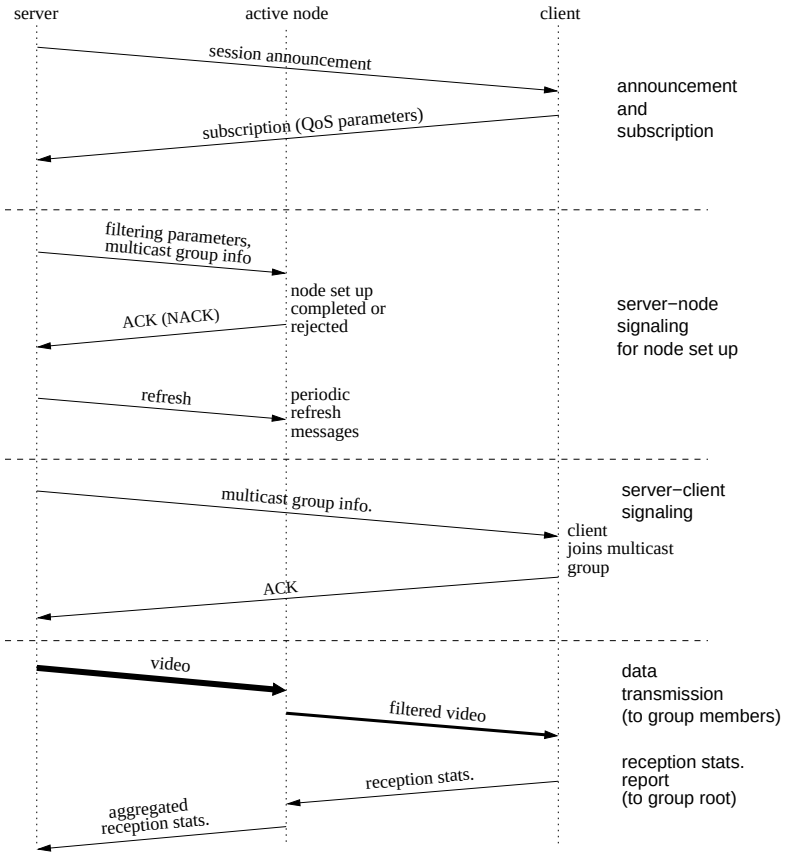

Fig. 3. Signaling required for the proposed framework

requested by the client reflects not only the user's preference on the perceived video quality but also limitations on its available resources [10]. 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*. As explained before, 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. It is possible that node set up fails due for example to

insufficient resources, and in this case, we must 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 before the start of the video transmission, set up fails if it is not able to do so.

We assume the use of soft state, it means that after set up is done, it is necessary to send "refresh" messages periodically to maintain the node waiting for packets to be processed by the filtering code. If no refresh messages are sent, it is assumed that filtering is no longer needed and the node releases the reserved resources.

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 [11]. 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 detail in this work, the nature of best-effort networks makes necessary to monitor the reception conditions of the clients, since usable bandwidth for the video session is not assured. Active nodes can be used to check this. The advantage of having a hierarchy of groups is that feedback implosion can be controlled. 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 streaming applications include packet loss, delay and jitter. We can use RTP packets [12] to send the data, and then infer those parameters. The utilization of this information, i.e., how to use it to dynamically modify the distribution tree, is left for future study.

3 Algorithm for Construction of the Multicast Distribution Tree

In this section we detail our approach for the construction of the multicast distribution tree previous to the start of the transmission. As described in Section 2, it consists of a hierarchical conformation of multicast groups, and the purpose of the algorithm is to

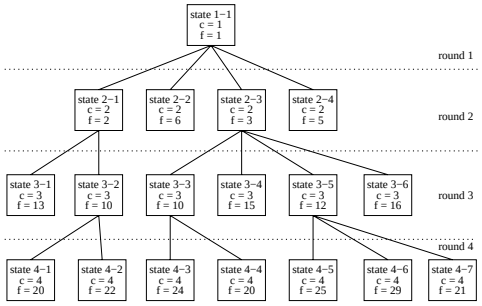

Fig. 4. Example of a state tree

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 therefore each client request can be expressed by a scalar value, that 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.

Our algorithm forms a distribution tree in a request by request basis, taking the requests in descending order of quality. In the case that there are many requests with the same quality, we take first the ones from the clients closer to the sender. We try to use the sender to stream to the clients that require the highest quality, and choose the nodes located in the best place to perform filtering. The designated active nodes becomes the root node of a new multicast group of a filtered video stream of lower quality. The filtered stream is then sent to clients that demanded lower quality streams.

Each step in the construction of the tree defines a state. The state is defined by a variable c that stands for the number of clients that have been already considered, and the characteristics of the distribution tree needed to serve those clients, that is, which nodes have been used to filter and produce, if any, the stream with the requested quality. Fig. 4 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 deriving states from a round in the state tree, we define a set of "candidate senders" to provide the requested stream to the client newly considered in the next round. Either the original server of the video sequence or any of the active nodes in the network can be the candidate sender. For a given flow request and candidate sender, one of the following situations is possible:

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.

The election of the filtering nodes is based:

1. On the distance, i.e., number of hops, between the client and the candidate node. The first candidate to choose is the closest one 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 ones are chosen close to this one.
2. On a function f that considers other factors such as total bandwidth used, link utilization, and/or the use of node resources. This function can be thought as a measure of how good is the complete distribution tree being formed. A lower value of f means a better distribution tree.

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.

As described above, our algorithm belongs to the category of exhaustive search algorithms. It means that the number of possible states in each round directly affects the efficiency of our algorithm. 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 new states generated 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 number of hops criterion explained above. 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.

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

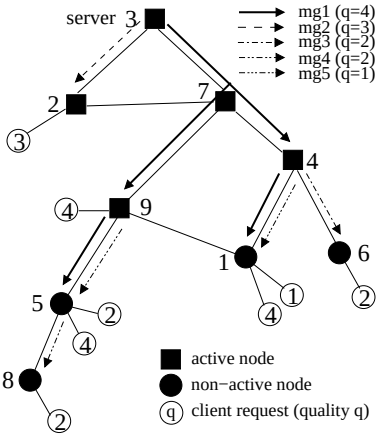

Fig. 5. Multicast groups for our algorithm

3.1 Example

Fig. 5 shows an example network topology 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 [13] for the MPEG-2 video coding algorithm [14]. In layered video case, 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.

Fig. 5 shows the multicast groups 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 1, just to provide filtered streams to clients in nodes 1 and 6.

4 Evaluation

In this section, we show the effectiveness of our proposed algorithm through some numerical experiments. We generate random topologies using Waxman's algorithm [15], and choose the parameters appropriately to generate topologies with an average degree of 3.5, to try to imitate the characteristics of real networks [16]. 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 operations that each active node can afford 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. We apply two other approaches for multicast tree construction to the same topologies for comparison purposes. Those are simulcast and distribution of layered coded video.

The definition of f, which is used to evaluate the effectiveness of the built tree in the algorithm 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, those are for minimizing bandwidth and minimizing link utilization, which we'll call f_1 and f_2:

$$f_1 = \sum_{i \in \mathcal{U}} B_i \tag{1}$$

$$f_2 = \frac{\sum_{i \in \mathcal{U}} B_i}{|\mathcal{U}|} \tag{2}$$

where i denotes a used link, $\mathcal{U}$ is the set of used links, and B_i denotes the used bandwidth in link i. With f_1 we wanted to minimize the total bandwidth used per session, and with f_2 we expected that our algorithm could perform some sort of "load balancing," to avoid congesting a single link.

We compare our algorithm increasing the number of sessions in the network to see how many sessions can be simultaneously set up and provided for users. In the experiments, all the links are assumed to have a bandwidth of 100 Mb/s. We multiplex sessions, each of which is set up according to our algorithm, 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 taken into account in our algorithm stated in Section 3. Thus, the number we consider here is that 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.

In Figs. 6 and 7, experiments 1-10, 11-20, 21-30, 31-40, 41-50, 51-60 refer to 20-nodes 10-requests, 20-nodes 20-requests, 20-nodes 50-requests, 50-nodes 10-requests, 50-nodes 20-requests, and 50-nodes 50-requests cases, respectively.

Fig. 6 shows the average bandwidth required to establish all 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 of used links in each session, when we sum all

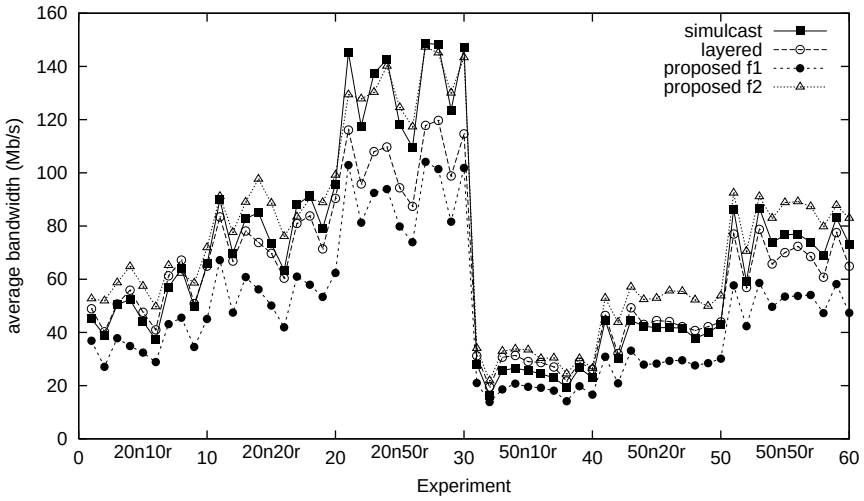

Fig. 6. Average bandwidth (Mb/s) for the first ten sessions

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.

Fig. 7 shows the maximum number of simultaneous sessions up to 15 that could be set up using each one of the three methods, i.e., the proposed algorithm, simulcast and layered distribution. 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 some few cases in which our proposed algorithm was surpassed by the layered video approach. We expect this to occur when we have the same stream with different qualities over the same link, congesting it as it occurs in simulcast. 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 due to space limitation. With f_2 we expected to increase the number of possible simultaneous sessions, reducing the bandwidth used per link, at the expense of increasing the number of used links. However, f_2 only increases greedily the number of used links in the tree, sometimes misplacing the filtering location.

5 Summary

We presented 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. In our approach, all the quality requests

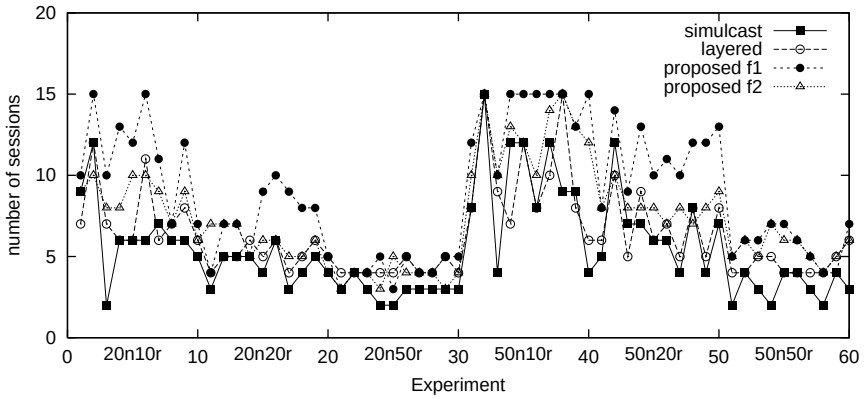

Fig. 7. Maximum number of simultaneous sessions (out of 15)

are collected and the video server infers a multicast distribution tree prior to the video transmission.

We then presented an algorithm for electing the filtering nodes in this distribution tree, which aims to minimize a function f that can be set to consider some network parameters, to achieve efficient use of the 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. We compared our algorithm with other two methods of distributing video that not consider the use of active nodes: simulcast and layered encoded distribution, and found that using our algorithm we can set up a greater number of simultaneous sessions, meaning a more effective use of the available bandwidth of the network, but at the expense of requiring processing capability at the network nodes.

Future research topics include: the election of other definitions for f to improve the distribution tree, consideration of the effect of delays introduced at the filtering nodes, and the analysis to use 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 Scientific Research (A) 11305030 and a Grant-in-Aid for Encouragement of Young Scientists 12750322 from The Ministry of Education, Science, Sports and Culture of Japan, Special Coordination Funds for Promoting Science and Technology of the Science and Technology Agency of the Japanese Government, and Telecommunication Advancement Organization of Japan under the Project “Global Experimental Networks for Information Society.”

References

1. McCanne, S., Jacobson, V., Vetterli, M.: Receiver-driven Layered Multicast. Proc. ACM Sigcomm (1996) 117-130
2. Yeadon, N., García, F., Hutchinson, D., Shepherd, D.: Filters: QoS Support Mechanisms for Multipeer Communications. IEEE Journal on Selected Areas in Communications, Vol. 14, No. 7 (1996) 1245-1262
3. Pasquale, J., Polyzos, G., Anderson, E., Kompella, V.: Filter Propagation in Dissemination Trees: Trading Off Bandwidth and Processing in Continuous Media Networks. Proc. NOSSDAV (1993)
4. Metzler, B., Harbaum, T., Wittmann, R., Zitterbart, M.: AMnet: Heterogeneous Multicast Services based on Active Networking. Proc. IEEE OpenArch (1999) 98-105
5. Calvert, K. (ed.): Architectural Framework for Active Networks, Version 1.0. Active Network Working Group Draft (1999)
6. Alexander, S., Arbaugh, W., Hicks, M., Kakkar, P., Keromytis, A., Moore, J., Gunter, C., Nettles, S., Smith, J.: The Switchware Active Network Architecture. IEEE Network, Vol. 12 No. 3 (1998) 27-36
7. Wetherall D.: Service Introduction in an Active Network. Ph.D. Thesis, Massachusetts Institute of Technology (1999)
8. Tanenbaum, A.: Computer Networks. 3rd. edn. Prentice Hall (1996)
9. Handley, M.: SAP: Session Announcement Protocol. Internet Draft, Work in Progress (1996)
10. Fukuda, K., Wakamiya, N., Murata, M., Miyahara, H.: On Flow Aggregation for Multicast Video Transport. Proc. Sixth IFIP International Workshop on Quality of Service (1998) 13-22
11. Semeria, C., Maufer, T.: Introduction to IP Multicast Routing. 3COM White Paper, available at `http://www.3com.com`
12. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RTP: A Transport Protocol for Real-Time Applications. Request for Comments 1889 (1996)
13. Fukuda, K., Wakamiya, N., Murata, M., Miyahara, H.: Real-Time Video Distribution with Hybrid Hierarchical Video Coding in Heterogeneous Network and Client Environments. Proc. MMNS (1998)
14. ISO/IEC DIS 13818-2: MPEG-2 Video. ISO Standard (1994)
15. Waxman, B.: Routing of Multipoint Connections. IEEE Journal on Selected Areas in Communications, Vol. 6, No. 9 (1988) 1617-1622
16. Zegura, E., Calvert, K., Bhattacharjee, S.: How to Model an Internetwork. Proc. IEEE Infocom (1996)