Receiver-Cooperative Bandwidth Management for Layered Multicast

Hirozumi Yamaguchi[†] Keiichi Yasumoto[‡] Teruo Higashino[†] Kenichi Taniguchi[†]
[†]Graduate School of Eng. Sci., Osaka University [‡]Faculty of Economics, Shiga University
Toyonaka, Osaka 560-8531, JAPAN Hikone, Shiga 522-0069, JAPAN
{h-yamagu, higashino, taniguchi}@ics.es.osaka-u.ac.jp yasumoto@biwako.shiga-u.ac.jp

Abstract

In this paper, we propose a receiver-cooperative bandwidth management method for layered multicast streams, considering not only bandwidth requirements but also receivers' preference. In our method, we assume that each receiver has a preference value for each layer of streams. When a receiver requests an additional layer and required bandwidth is not available on links, it can let other receivers release a part of the layers of streams which they receive if, it increases the sum of the satisfied preference values of all receivers as a total. We give an algorithm to calculate an optimal way of releasing layers in a polynomial time under some assumption. We had an experiment to transmit JPEG video over a private IP network and confirmed that our method could control bandwidth among the video streams within a second.

1 Introduction

In multicast communication[1], receivers are often located on several nodes over internetworks, where available bandwidth at each node is different from each other. Therefore, it is difficult for a sender to control its transmission rate. For example, if a sender adapts its transmission rate to the receivers on low bit-rate networks, it cannot provide high quality video to the receivers on high bit-rate networks. On the other hand, if it adapts the transmission rate to the receivers on low bit-rate networks, each receiver on low bit-rate networks may lose some of the video packets.

For this problem, there have been proposed some receiver-driven bandwidth management methods where rate adaptation has migrated from a sender to receivers. Simulcasting is known as one of such methods, where a video is encoded at several different quality levels, and delivered as several streams. Each receiver selects only one of the streams to receive, depending on its available bandwidth [10]. Layered multicasting with hierarchical encoding provides a more efficient way, where a video is encoded into a set of layers, one basic layer and some extended layers. The basic layer contains the indispensable data for decoding, and each extended layer has the remainder to improve the quality of the video. The l-th layer has the data which can further improve the quality of the video decoded from the 1st layer (the basic layer) and the 2nd, ... and $(l-1)$-th layers (the lower extended layers).

Each receiver can receive the basic layer and the adequate number of extended layers depending on its available bandwidth [2, 3, 4].

In the receiver-driven bandwidth management based on hierarchical encoding, an adequate quality is offered according to each receiver's available bandwidth. However, in a session of a video conference, the bandwidth reserved for the session is used by several multicast streams from multiple senders (*e.g.*, the members of an assembly) simultaneously. So these streams may compete with each other for the limited bandwidth. Moreover, we have to consider that each receiver has his/her own preference for the streams (*e.g.*, some receivers may want to see only the chairman's video with high quality, while others may want to see all the members' videos with intermediate quality). Such preference may change when the current subject in the assembly changes. Thus, in the session with the limited bandwidth, it is desirable that bandwidth is dynamically controlled among the streams so that the satisfaction of receivers' preference is maximized, and it should be done in a receiver-oriented fashion since each receiver's preference is different from the others.

In this paper, we propose a receiver-cooperative bandwidth management method called RECOMM where a request of a receiver for receiving an additional layer of a stream can be accommodated even when the current available bandwidth is insufficient, by letting other receivers release a part of the layers of streams which they receive. We assume that each receiver gives in advance a "preference value" for each layer of the streams. When a receiver requests an additional layer, it calculates such a set of layers received by other receivers that minimize the sum of preference values for the layers. Then the request is accom-

modated if the preference value for the requested layer is greater than the calculated sum of the preference values. Our algorithm can find such an optimal set of layers in a polynomial time with respect to the number of streams and path length if each path between any two routers on multicast routing trees can uniquely be decided. This condition always holds for the case that multicast routing trees form a shared tree like Core Based Tree (CBT) routing[1]. We have implemented RECOMM and carried out an experiment to transmit three JPEG video streams, each of which is divided into four layers, via IP multicast tunneling on a private IP network. We have confirmed that RECOMM could control the bandwidth of the streams within a second. WE have also carried out simulation to show the efficiency of RECOMM.

This paper is organized as follows. In Section 2, the outline of RECOMM is summarized. In Section 3, the optimal layer releasing problem is formulated. The algorithm for calculating an optimal set of layers is given in Section 4. Section 5 gives experimental results. Section 6 concludes this paper and explains how we give preference values for video conferences.

1.1 Related Work

The paper [4] gives a receiver-driven rate adaptation method based on layered multicast, on a network whose topology is unknown. It decides the priority of the layers of streams by collecting and calculating the receivers' preference values. In [5], Shacham, et. al. have proposed a method for deciding the number of layers of streams which each receiver can receive, based on each receiver's preference values and link capacity constraints. In this method, the number of the layers received by each receiver is decided so that the sum of all receivers' preference values (they call it "bid") is maximized (or nearly maximized). However, they have a restriction that all streams must use the same multicast tree from exactly one source host. Therefore, it is not so easy to apply this method to video conferences where a lot of senders and receivers are widely located over a network, or video-on-demand systems where multimedia resources are placed on different locations.

Also, in such applications, receivers' requirements may change dynamically after session establishment. Some methods based on "bandwidth preemption"[6, 7] are applicable to such situations, which allow preemption from existing streams of some receivers. However, [6] allows bandwidth preemption only among one receiver's layers, and [7] does not consider each receiver's preference. Our method can manage bandwidth dynamically among multiple multicast streams from different sources by cooperation of receivers, based on receivers' preference for the streams.

2 RECOMM Overview

2.1 Layered Multicast

A network is modeled as a set of *multicast routers* (or *routers* in short) and links between routers. Each link has a capacity called a session capacity. Each of senders or receivers is assumed to be located on LAN connected to one of the routers. A tree structured data stream from one sender S_i ($1 \leq i \leq M$) to some receivers R_1, ..., R_N is called a *multicast stream* (or *stream* in short) and denoted by st_i. We assume that each stream st_i is hierarchically encoded[2, 3] into L layers, a basic layer (the first layer) and $L-1$ extended layers (the second, third, ... and ($L-1$)-th layers). $\langle st_i, l\rangle$ denotes the l-th layer of the stream st_i. The hierarchically encoded stream st_i can be decoded using only its basic layer, however, if we can use some of the extended layers, we can decode st_i with higher quality. An extended layer $\langle st_i, l\rangle$ contains additional data to improve the quality of the stream st_i decoded from $\langle st_i, 1\rangle$, ... and $\langle st_i, l-1\rangle$. Delivering such layers by independent multicast addresses enables each receiver to select the number of the layers to receive, according to its own available bandwidth. Hereafter, we denote the bandwidth necessary for receiving the layer $\langle st_i, l\rangle$ by $B\langle st_i, l\rangle$.

2.2 Bandwidth Management by RECOMM

Assume that a receiver R_Y which has received $\mathcal{L}-1$ layers of a stream st_X ($\langle st_X, 1\rangle$, ... and $\langle st_X, \mathcal{L}-1\rangle$) requests for receiving the upper layer $\langle st_X, \mathcal{L}\rangle$ in order to improve its quality. Let RT denote the last router on the path from S_X to R_Y, which relays $\langle st_X, \mathcal{L}\rangle$. In most bandwidth reservation protocols based on the general QoS control policies, the requirement for the additional bandwidth is accommodated and the bandwidth is reserved only when at least $B\langle st_X, \mathcal{L}\rangle$ bandwidth is left unused on every link on the path from RT to R_Y. However, if some links do not satisfy this condition, the request is not accommodated in order to keep the quality of the existing streams.

We propose a method called RECOMM (REceiver-COoperative Multicast bandwidth Management), in order to accommodate such a request if it increases the satisfied preference of receivers as a total. In RECOMM, even in the case above, R_Y may obtain bandwidth by letting some other receivers release a part of the layers of streams which they receive. For this purpose, we assume that each receiver R_j gives its own preference for each layer $\langle st_i, l\rangle$ as a positive real number in advance, called a *preference value* (denoted by $W_j\langle st_i, l\rangle$). Such a set of layers of the other receivers that makes at least $B\langle st_X, \mathcal{L}\rangle$ bandwidth unused on each link from RT to R_Y and minimizes the sum of the preference values for the layers is calculated by R_Y. R_Y can

Table 1. Notation.

Symbols	Contents
S_i	i-th sender ($1 \leq i \leq M$)
R_j	j-th receiver ($1 \leq j \leq N$)
st_i	multicast stream from S_i
$tree_i$	the set of links on the routing tree of st_i
$path_{i,j}$	the set of links in $tree_{i,j}$ on the path from S_i to R_j
$\langle st_i, l \rangle$	l-th layer of stream st_i
$W_j\langle st_i, l \rangle$	the preference value of receiver R_j for layer $\langle st_i, l \rangle$
$B\langle st_i, l \rangle$	the bandwidth necessary for delivering layer $\langle st_i, l \rangle$

let the corresponding receivers release the layers only if the preference value $W_Y\langle st_X, \mathcal{L} \rangle$ is greater than the calculated sum of the preference values for the layers. Note that the notations are summarized in Table 1.

2.3 Example Case

Fig. 1 shows a situation that receivers R_1, R_2, R_3 and R_4 receive some layers of streams st_1, st_2 and st_3, where each stream is encoded into three layers. For example, receiver R_1 belonging to router RT_6 receives the first and second layers of st_1.

Suppose that the session capacities of the links RT_3-RT_8, RT_8-RT_9 and RT_9-RT_{10} are 20, 19 and 19, respectively. Also suppose that receiver R_4 belonging to RT_{10} has just requested the second layer of st_3 $\langle st_3, 2 \rangle$ for improving the quality of st_3. However, on the link RT_3-RT_8 for example, there already exist three streams st_1, st_2 and st_3 which consume $6 + 4 = 10$, $3 + 3 = 6$ and 4 units of bandwidth, respectively. Therefore, its current unused bandwidth is 0 and the required bandwidth $B\langle st_3, 2 \rangle = 3$ cannot be obtained. Then R_4 tries to find such a set of layers of R_1, R_2, R_3 and R_4 that (a) generates at least three units of unused bandwidth on each link and (b) minimizes the sum of the preference values of the receivers for the layers.

Fig. 2 shows the preference values of the receivers, corresponding to Fig. 1. Each dotted layer means that it is currently received by a receiver. In this situation, letting R_1, R_2, R_3 and R_4 release the second and third layers of st_1 is the way to minimize the sum of the preference values ($W_1\langle st_1, 2 \rangle + W_2\langle st_1, 2 \rangle + W_2\langle st_1, 3 \rangle + W_3\langle st_1, 2 \rangle + W_4\langle st_1, 2 \rangle + W_4\langle st_1, 3 \rangle = 2+2+2+4+1+3 = 14$), and to make at least three units of bandwidth unused on links RT_3-RT_8, RT_8-RT_9 and RT_9-RT_{10}. Therefore, if R_4 requires $\langle st_3, 2 \rangle$ with the preference value $W_4\langle st_3, 2 \rangle = 15 > 14$, this request is accommodated by letting R_1, R_2, R_3 and R_4 release the second and third layers of st_1.

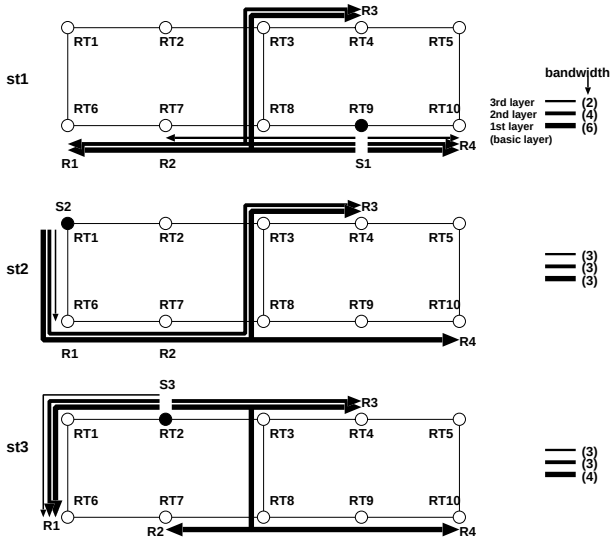

Figure 1. Layered mutlicast streams.

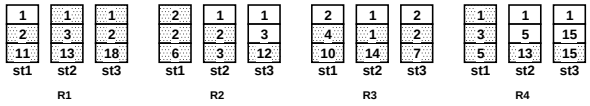

Figure 2. Preference values.

3 Optimal Layer Releasing Problem

We call the problem to find such a set of layers that minimizes the preference values "optimal layer releasing problem". In this section, we formulate the optimal layer releasing problem.

Hereafter, on the path from sender S_X to receiver R_Y which requests layer $\langle st_X, \mathcal{L} \rangle$, we denote the links from the last router RT which relays layer $\langle st_X, \mathcal{L} \rangle$ to R_Y, by L_1, ..., L_P, respectively. For example, for the request of the additional layer $\langle st_3, 2 \rangle$ by R_4 in Fig. 1, links RT_3-RT_8, RT_8-RT_9 and RT_9-RT_{10} correspond to L_1, L_2 and L_3, respectively. We also use the following notations.

- $LR_{i,j}$: the number of layers of st_i which receiver R_j currently receives (given)
- $lrnew_{i,j}$: the number of layers of st_i which receiver R_j will receive after layer releasing

For given (a) session bandwidth on each L_k ($k = 1, ..., p$) (denoted by $C(L_k)$), (b) each $path_{i,j}$ which shares at least one link with the set of links L_1,...,L_P and (c) $LR_{i,j}$, $W_j\langle st_i, l \rangle$ and $B\langle st_i, l \rangle$ ($l = 1, ..., L$) for each pair of i and j of (b), the optimal layer releasing problem is to find a set of $lrnew_{i,j}$ which minimizes the following objective func-

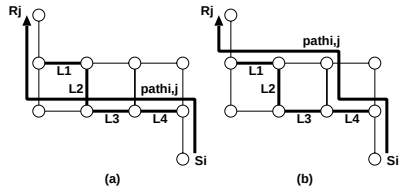

Figure 3. $path_{i,j}$ and links L_1,L_2,L_3 and L_4.

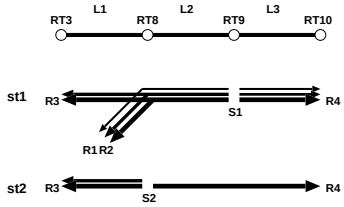

Figure 4. Delivering of st_1, st_2 on links RT_3-RT_8-RT_9-RT_{10}.

tion:

$$\sum_j \sum_i \sum_{lrnew_{i,j} < l \le LR_{i,j}} W_j \langle st_i, l \rangle \quad (1)$$

subject to:
$\forall k$

$$C(L_k) - \sum_i \sum_{1 \le l \le llnew_{i,k}} B\langle st_i, l \rangle \ge B\langle st_X, \mathcal{L} \rangle \quad (2)$$

$\forall h, k\ (1 \le h < k \le p,\ L_h, L_k \in tree_i)$

$$llnew_{i,h} \ge llnew_{i,k} \quad (3)$$

where we assume that L_h is an upstream link of L_k on $tree_i$ without loss of generality. Note that $llnew_{i,k}$ is defined as follows.

$$llnew_{i,k} \stackrel{def}{=} max_{j:L_k \in path_{i,j}} \{lrnew_{i,j}\} \quad (4)$$

Function (1) represents the sum of the preference values for the layers released. Here, $llnew_{i,k}$ denotes the number of layers of st_i still delivered through L_k after layer releasing. Therefore, constraint (2) represents that bandwidth left unused on L_k should be at least $B\langle st_X, \mathcal{L} \rangle$. Constraint (3) represents that the number of layers on an upstream link after layer releasing should be equal to or bigger than that on its any downstream link.

R_Y compares the minimized value of Function (1) with the value of $W_Y\langle st_X, \mathcal{L} \rangle$, and lets the corresponding receivers release the layers only if the latter is bigger than the former.

4 Algorithm for Optimal Layer Releasing

The algorithm given in this section for the optimal layer releasing problem works within a polynomial time if the set of links shared by each $path_{i,j}$ and the set of links $\{L_1$, ..., $L_P\}$ form subsequential links. For example, in Fig. 3(a), $path_{i,j}$ and links L_1,...,L_4 have common subsequential links L_3 and L_4, while L_1 and L_4 in Fig. 3(b) are not subsequential links. For the case that the condition does not hold, we will give discussion in Section 4.2.

In practice, one of the useful sufficient conditions for the restriction mentioned above is, for any pair of routers RT_u and RT_v, the routing protocol can uniquely decide a path between them. For example, a routing protocol based on Core Based Tree forms a shared tree for all the multicast streams, and this condition always holds. On the other hand, if the routing protocol determines each routing tree based on link costs (*e.g.* hop counts) such as DVMRP, this condition may not hold, since there may exist several paths of the same cost.

4.1 Polynomial Time Algorithm

The basic idea of the algorithm is as follows. On all the links L_1, ..., L_P, the algorithm finds the optimal sets of the top layers of streams satisfying the following conditions: (a) their total amount of bandwidth on each link is bigger than 0, and (b) the sum of the preference values for the layers per unit of the bandwidth is minimum of all the possible sets of top layers. Let $CAP_{1,P}$ and $PREF_{1,P}$ denote the minimum bandwidth obtained by releasing the set of the layers through the links L_1, ..., L_P and the sum of the preference values for the layers, respectively (thus we find the set of top layers where $\frac{PREF_{1,P}}{CAP_{1,P}}$ is minimum). Such a set of top layers can be considered as a candidate to be released, since the sum of their preference values is minimum of all the possible sets of top layers. The same procedure is applied to the remained layers after excluding the candidate and is repeated until the required bandwidth is obtained.

To formalize the above procedure, we define each set of receivers $rset_{k,h}\langle st_i, l \rangle$ for each top layer, where the delivery of the top layer is stopped on L_k, ..., L_h if all these receivers release the layer. We also define $pref_{k,h}\langle st_i, l \rangle$ as the sum of the preference values of these receivers in $rset_{k,h}\langle st_i, l \rangle$ for the layer $\langle st_i, l \rangle$. We find all the possible $rset_{k,h}\langle st_i, l \rangle$ and $pref_{k,h}\langle st_i, l \rangle$. Bandwidth left on each link is represented as a virtual stream consisting of only one layer for the convenience. $\langle st_0, 1 \rangle$ denotes such virtual stream where $rset_{k,k}\langle st_0, 1 \rangle = \emptyset$ and $pref_{k,k}\langle st_0, 1 \rangle = 0$ for each L_k.

In order to find the optimal set of top layers, we must check all the combination of the set of top layers in general cases (it may cause exponential explosion in the worst case). Now suppose that $RSET_{k,h}$ denotes such an optimal set of top layers on links L_k, ..., L_h. By the restriction about

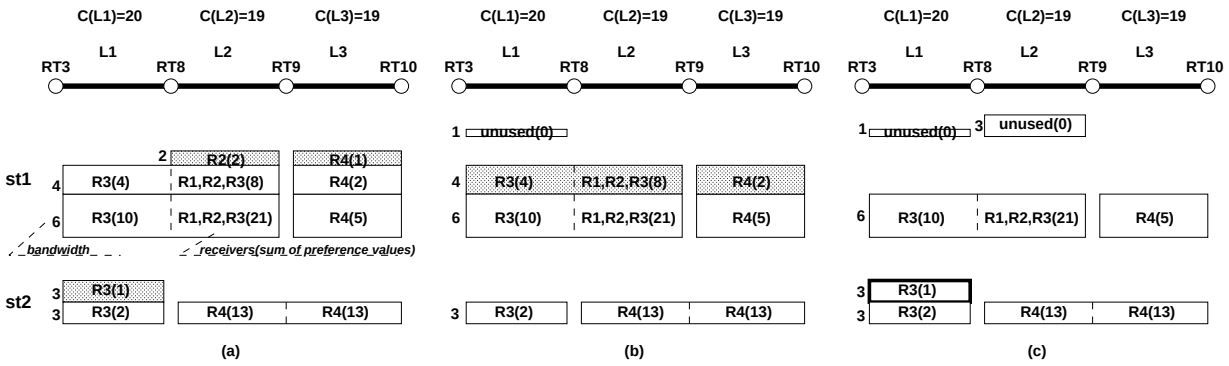

Figure 5. Receivers, preference values and bandwidth for layers of st_1 and st_2 on links L_1, L_2 and L_3.

shared links (mentioned above), our algorithm can decide the optimal set $RSET_{1,z}$ from $RSET_{1,1}$, $RSET_{1,2}$, ... and $RSET_{1,z-1}$. This is the keypoint that our algorithm works in a polynomial time. The complexity analysis is given later.

Fig. 4 shows the layers of streams st_1 and st_2 delivered on links RT_3-RT_8, RT_8-RT_9 and RT_9-RT_{10} (denoted by L_1, L_2 and L_3, respectively) in the example in Section 2.3. Fig. 5(a) also shows the layer delivery on these links in Fig. 4. Here, each box corresponds to a layer and a string inside like "R_1,R_2,R_3(8)" denotes receiver' names and the sum of their preference values for the layer. Each number beside a box shows bandwidth necessary for the layer. Starting from Fig. 5(a), we can easily find

- $RSET_{1,1} = \{rset_{1,1}\langle st_2, 2\rangle\}$,
- $RSET_{1,2} = \{rset_{1,1}\langle st_2, 2\rangle, rset_{2,2}\langle st_1, 3\rangle\}$ and
- $RSET_{1,3} = \{rset_{1,1}\langle st_2, 2\rangle, rset_{2,2}\langle st_1, 3\rangle, rset_{3,3}\langle st_1, 3\rangle\}$,

which correspond to the dotted boxes in Fig. 5(a).

Since $CAP_{1,3} = 2$ and $B\langle st_2, 2\rangle = 3$, unused bandwidth 1 appears on L_1 on the next (second) stage. Fig. 5(b) shows the second stage, where we find $RSET_{1,1} = \{rset_{1,1}\langle st_0, 1\rangle\}$ since its preference value is 0. Then, in order to find $RSET_{1,2}$, we should compare (a) $RSET_{1,1} \cup \{rset_{2,2}\langle st_2, 1\rangle\}$ (capacity through L_1 and L_2 is 1), (b) $RSET_{1,1} \cup \{rset_{2,2}\langle st_1, 2\rangle\}$ (capacity 1) and (c) $\{rset_{1,2}\langle st_1, 2\rangle\}$ (capacity 4). In this case, (c) is feasible since the sums of the preference values per unit of bandwidth for the three sets are, (a) $(0 + 13)/1 = 13$, (b) $(0 + 8)/1 = 8$ and (c) $8/4 = 2$, respectively. Thus $RSET_{1,2} = \{rset_{1,2}\langle st_1, 2\rangle\}$. Finally, in order to find $RSET_{1,3}$, we compare (a) $RSET_{1,1} \cup \{rset_{2,3}\langle st_2, 1\rangle\}$ (capacity 1), (b) $RSET_{1,2} \cup \{rset_{3,3}\langle st_1, 2\rangle\}$ (capacity 4) and (c) $RSET_{1,2} \cup \{rset_{3,3}\langle st_2, 1\rangle\}$ (capacity 3). Since the sums of the preference values per unit of bandwidth for the above sets are $(0 + 13)/1 = 13$, $(8 + 2)/4 = 2.5$ and $(8 + 13)/3 = 7$, respectively, we find $RSET_{1,3} = RSET_{1,2} \cup \{rset_{3,3}\langle st_1, 2\rangle\}$ and $CAP_{1,3} = 4$. At the end of this stage, the total capacity is $2+4 = 6$. Now we stop to find layers. Note that we may have to reduce the redundant layers. We checked for each entry in

$$RSET = \{rset_{1,1}\langle st_2, 2\rangle, rset_{2,2}\langle st_1, 3\rangle, rset_{3,3}\langle st_1, 3\rangle, rset_{1,2}\langle st_1, 2\rangle, rset_{3,3}\langle st_1, 2\rangle\}.$$

We find that $rset_{1,1}\langle st_2, 2\rangle$ is not necessary since $rset_{1,2}\langle st_1, 2\rangle$ covers L_1 with 4 units of bandwidth (the required bandwidth is 3 on each link). Finally, we decide

$$RSET = \{rset_{2,2}\langle st_1, 3\rangle, rset_{3,3}\langle st_1, 3\rangle, rset_{1,2}\langle st_1, 2\rangle, rset_{3,3}\langle st_1, 2\rangle\}$$

as shown in Fig. 5(c).

4.2 Complexity Analysis

Suppose that layer $\langle st_i, l\rangle$ is the top layer on L_x, ..., L_y, and st_i is delivered in the direction from L_x to L_y. If its delivery is stopped on L_k ($x \le k \le y$), it is also stopped on all the down stream links L_{k+1}, ..., L_y. Therefore the total number of $rset$s regarding a top layer $\langle st_i, l\rangle$ of st_i is at most $y-x$ ($\le P$) where P is the number of links. As a total, the number of possible $rset$s for top layers of streams is less than MP, where M is the maximum number of streams. Therefore, we can find each $RSET_{1,z}$ by at most MP times of comparisons. Since z ranges from 1 to P, the time complexity to find each set of top layers is $O(MP^2)$ in the worst case. After finding a set of top layers, the delivering of at least one top layer of a stream is stopped. Thus in the worst case, the procedure to find a set of top layers is executed

Table 2. JPEG video.

Codec	JPEG (inter-frame independent)
Resolution	160x120 (5 kbyte/frame)
Layer	4: 32fps (600 kbps) 3: 16fps (300 kbps)
	2: 8fps (150 kbps) 1: 4fps (150 kbps)

MPL times, the maximum number of layers on L_1, ... and L_P. Note that L is the maximum number of layers of a stream. In order to exclude unnecessary $rset$s from $RSET$, we only checks at most MP $rset$s in $RSET$. Here, we need P times of comparisons for each $rset$. Therefore, its complexity is $O(MP^2)$. As a result, the time complexity of the algorithm is $O(MP^2 * MPL) = O(M^2P^3L)$, where M, P are L are the numbers of streams, links and layers, respectively (L can be regarded as a constant in general).

For some streams which do not satisfy the restriction, we use the modified algorithm where the possible combinations of $rset$s for top layers of streams may have to be checked to decide each $RSET_{1,P}$, since each $rset$s has to be constructed for possible subsets of links L_1, ... and L_P (*i.e.*, we need to define variables like $rset_{1,3,5,7}\langle st_i, l\rangle$). Its time complexity is $O(P^M)$ ($O(P^{M+1}ML)$ as a total) if all the streams do not satisfy the restriction. However, if the number of such streams is small and can be considered as a constant, the time complexity would still be within polynomial time.

5 Experimental Results

5.1 JPEG Video Transmission over IP Network

We have implemented RECOMM and a program to transmit a layered JPEG video stream via multiple multicast addresses. We have also constructed a private IP network and had an experiment to deliver layered JPEG video streams and control bandwidth among them.

We have implemented RECOMM as a set of modules, each of which corresponds to each receiver. We denote RECOMM module of receiver R_j by RECOMM$_j$. RECOMM$_j$ keeps information about the streams which R_j currently receives (the preference values for them, path from the sources and so on) and collect information from other modules when it calculates the optimal layer releasing. RECOMM modules communicate with each other via a multicast address.

The network is constructed by connecting nine IP subnetworks on 10 Mbps Ethernet via seven PCs (Pentium 150 MHz with 64 Mbyte Memory, FreeBSD 3.1). An implementation of DVMRP and IGMP, "mrouted", works on four of the seven hosts (see Fig. 6(a)), connected with each other via IP tunneling. There exists a pair of a video server S_i and a receiver R_i on each local subnets. On this network, each

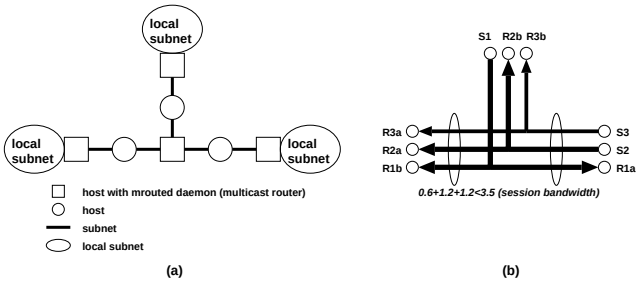

Figure 6. IP network and JPEG video streams.

server separates the frames of a JPEG video into four layers (Table 2) and transmits them via four multicast addresses. It consumes 1.2 Mbps when it is transmitted as the full motion video (32 fps by receiving full layers). The video playback facility has been implemented in the receiver program based on the XAnim source code. To generate network congestion explicitly, we have limited the maximum session bandwidth to 3.5 Mbps, using the facility of mrouted (thus the capacity of each link is regarded as 3.5 Mbps).

Each receiver (RECOMM module) tried to add layers from first to fourth step by step. Suppose the situation that the full layers of st_1 and st_2 and three layers of st_3 are being delivered (Fig. 6(b)). In this situation, RECOMM$_1$ tried to receive the fourth layer of st_3 by letting RECOMM$_1$ and RECOMM$_3$ to release the layers $\langle st_2, 4\rangle$ and $\langle st_1, 4\rangle$, respectively.

In this experiment, we have measured the change of frame rates of the streams st_2 and st_3 at RECOMM$_1$ shown in Fig. 7. RECOMM$_1$ started to receive $\langle st_3, 4\rangle$ at time 2 (sec) and it caused congestion. RECOMM$_1$ detected congestion at time 6, released $\langle st_3, 4\rangle$, collected information from RECOMM$_2$ and RECOMM$_3$, and let itself and RECOMM$_3$ release the layers $\langle st_2, 4\rangle$ and $\langle st_1, 4\rangle$, respectively. RECOMM$_1$ again started to receive $\langle st_3, 4\rangle$ at time 7 and it was accommodated normally. From the result, we have confirmed that RECOMM could control bandwidth within 1 second, not including the time to detect congestion.

5.2 Simulation

5.2.1 Scalability Analysis

We have developed a simulation tool on parallel programming language PARSEC[9], developed by UCLA Parallel Computing Laboratory, and had another experiment.

For three kinds of networks ((a) ring, (b) mesh and (c) bottleneck), we generated routing trees with minimum hop counts from a sender to each receiver, based on DVMRP. Senders and receivers are randomly located and the maximum session bandwidth on all the links are limited to 30

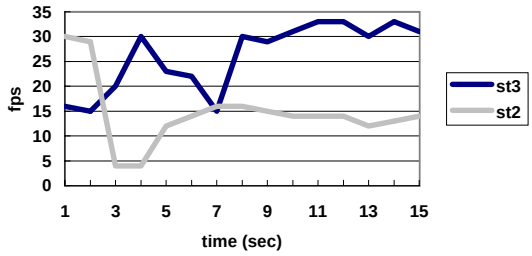

Figure 7. Frame rates of st_2 and st_3 at receiver R_1.

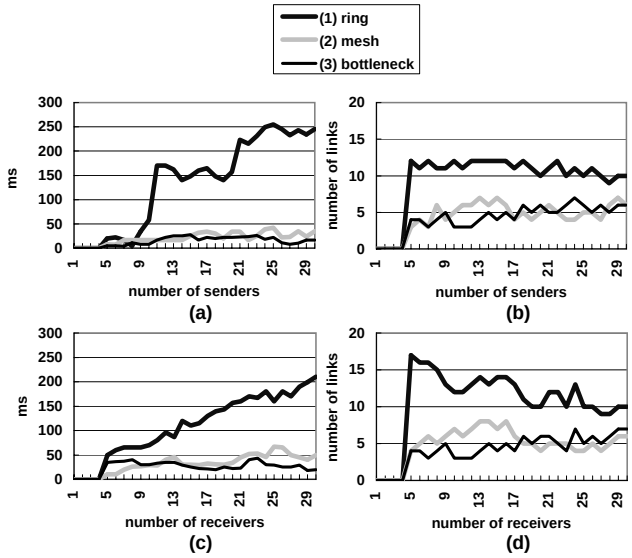

Figure 8. Computation time and number of links vs. numbers of senders and receivers.

(Mbps). We also assume high resolution MPEG2 video which consumes 12 Mbps when transferred as full motion video, containing a basic layer (5 Mbps) and two extended layers (4 Mbps and 3 Mbps). This experiment has been done on Enterprise 3000 of Sun Microsystems (Ultra Sparc, 512 Mbyte main memory, Solaris 2.6).

In the experiment, we began with measuring the response time for the request of an additional layer.

The response time T is the sum of T' and T'', where T' and T'' are computation time of optimal layer releasing and total time of messaging delay, respectively.

T' depends on the number of streams M and the number of links P where bandwidth should be reserved. P also depends on M and N, where N is the number of receivers. In the experiment, for $N = 20$ ($M = 20$) on networks with 50 routers, we had measured the average values of T' and P, for every M (N) ranging from 5 to 30. Fig. 8(a) and (b) show the average values of T' and P for M. Fig. 8(c) and (d) also show the average values of T' and P for N.

Table 3. Accommodation ratio.

	(X)	(Y)	(Z)
(i) ratio refused by bandwidth constraints	0.0%	36.4%	0.0%
(ii) ratio refused by preference value constraints	58.0%	38.4%	50.5%
(iii) total refusal ratio	58.0%	74.8%	50.5%

Fig. 8(d) shows that as the number of receivers N increases, the number of P decreases on ring networks. In general, the probability that some receivers have already received the streams becomes high if N is large on ring networks. Thus, the distance P between the last router RT which relays a layer of a stream and a receiver becomes shorter. This may prevent T' from growing up rapidly. On the other networks, P grows slowly. From these results, we have confirmed that our algorithm is scalable towards the increase of N, which may increase in proportion to the size of a session.

On the other hand, the total time of messaging delay T'' depends on the sum of the hop counts H of the messages. In general, the maximum hop count between receivers is, at most half of the number of routers for example in ring networks. Therefore we can estimate that H is at most twice as many as the number of routers (in this experiment, 100).

If we assume that the delay between routers is 10 milliseconds (this is large enough on assuming high speed networks between routers and the size of control messages), $T'' <= 1000$(ms). Since we know $T' < 250$(ms) in the experiment, the average response time T is less than 1.25 second. The maximum computation time was 1003 millisecond on the meshed network. Even if in such a case, we can get the response about 2 seconds. The result shows that the response time is reasonably short, considering time intervals while receivers' interests change in RECOMM applications. For example, a discussion subject may be changed in every few tens of minutes in a video conference.

5.2.2 Comparison

We did another experiment to show that our method can accommodate a request for an additional layer in a high probability. We have implemented the following three methods: (X) a heuristic method where layer releasing is computed on every link independently of the others, (Y) a method to decide an optimal layer releasing among the streams which only one receiver receives, and (Z) our method. We measured the ratio of the refused requests to all the requests. Each receiver sets a preference value randomly for a new layer which they would request. A receiver who issues a request or spontaneously releases a layer is also randomly

chosen for every certain time interval. 20 senders and 20 receivers are located on a ring network. The result is shown in Table 3. Here, (i) is the ratio of the requests refused by the reason that there is not enough bandwidth to be taken (this is effective only in (Y), since (Y) does not take bandwidth from the stream of other receivers). (ii) is the ratio of the requests refused by the reason that the sum of the preference values for the layer to be released is bigger than the preference value for a new layer. The local optimization method (Y) was refused by bandwidth constraints for many cases, since it only allows layer releasing among streams of one receiver. The heuristic method (X) was refused by preference value constraints, since it does not optimize the decrease of preference values. Our method (Z) achieved higher accommodation ratio (thus low refusal ratio) as a total.

6 Conclusion and Discussion

In this paper, we have proposed a bandwidth management method called RECOMM for layered multicast streams which allows to accommodate a receiver's request of a new layer, by letting other receivers release a part of their layers, when unused bandwidth is not sufficient. In the method, some receivers' satisfaction is reduced, however the decrease of their preference is minimized and the preference of all the receivers increases as a total. We have confirmed the efficiency of RECOMM through some experiments.

In practical applications, how each receiver gives preference values to layers should be considered. In a session of video conferences, for example, we can allow each receiver R_j to give any values to his/her layers unless their sum exceeds an upper bound C_j. By giving $C_1 = C_2 = ... = C_N$, all receivers will be fair. We can give different values for $C_1, ..., C_N$ based on some criteria such as the contribution to the session so that some particular persons can have higher priorities. In Fig. 2, we have defined $C_1 = C_2 = C_3 = C_4 = 50$.

Each receiver can easily distribute C_j among streams based on his/her preference for them (*e.g.*, if $C_j = 100$, and the receiver's preference to st_1, st_2 and st_3 are 10%, 50% and 40%, respectively, then $\sum_l w_j\langle st_1, l\rangle = 10$, $\sum_l w_j\langle st_2, l\rangle = 50$ and $\sum_l w_j\langle st_3, l\rangle = 40$). On the other hand, for the distribution of a value among layers of a stream, each receiver should know how much each layer contributes to increase the total quality of the stream. Thus some criteria should be given to receivers. In [7], we have given criteria for MPEG1 video streams based on their temporal resolution.

Our future work is as follows: (1) to design and implement a scheme with its user interface which automatically generates preference values from abstract user requirement by analyzing MPEG video codec with respect to the relation between the abstract user requirements (such as smoothness and size) and codec parameters (such as SNR); (2) to extend the proposed method to treat other network factors such as maximum delay.

Acknowledgement

The author Hirozumi Yamaguchi should be thankful to Professor Gregor von Bochmann at University of Ottawa, for his useful comments regarding this research and kind support while the author was a researcher at University of Ottawa in 1998.

References

[1] C. Diot, W. Dabbous and J. Crowcroft, "Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms," *IEEE Journal on Selected Areas in Communications*, Vol. 15, No. 3, pp. 277-290, April 1997.

[2] M. Ghanbari, "Two-Layer Coding of Video Streams for VBR Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 7, No. 5, pp.771–781, 1989.

[3] N. Shacham, "Multipoint Communication by Hierarchically Encoded Data," *Proc. of INFOCOM'92*, pp. 2107–2114, 1992.

[4] S. McCanne and V. Jacobson, "Receiver-driven Layered Multicast," *ACM SIGCOMM '96*, 1996.

[5] N. Shacham and H. Yokota, "Admission Control Algorithms for Multicast Sessions with Multiple Streams," *IEEE Journal of Selected Areas in Communications*, Vol. 15, No. 3, pp. 557-566, 1997.

[6] N. Shacham, "Preemption-Based Admission Control in Multimedia Multiparty Communications," *Proc. of INFOCOM'95*, pp.827–834, 1995.

[7] H. Sakate, H. Yamaguchi, K. Yasumoto, T. Higashino and K. Taniguchi, "Resource Management for Quality of Service Guarantees in Multi-party Multimedia Application," *the IEEE 1998 Int. Conf. on Network Protocols (ICNP'98)* , pp. 189–196, 1998.

[8] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu and L. Wei, "An Architecture for Wide-Area Multicast Routing," *Proc. of ACM SIGCOMM'94*, pp. 126-135, 1994.

[9] R. A. Meyer, "Parsec User Manual Release 1.1," *Online Document*, http://pcl.cs.ucla.edu/projects/parsec/, 1998.

[10] S. Fischer, A. Hafid, G.v. Bochmann and H.d. Meer, "Cooperative QoS Management for Multimedia Applications," *Proc. of Int. Conf. on Multimedia Computing and Systems (ICMCS'97)*, pp. 303–310, 1997.