

Resource Management for Quality of Service Guarantees in Multi-party Multimedia Application

Hiroharu Sakate Hirozumi Yamaguchi
Dept. of Info. and Math. Science
Osaka University
Toyonaka Osaka 560-8531 JAPAN
{sakate, h-yamagu}@ics.es.osaka-u.ac.jp

Keiichi Yasumoto
Dept. of Info. Processing and Management
Shiga University
Hikone Shiga 522-0069 JAPAN
yasumoto@biwako.shiga-u.ac.jp

Teruo Higashino Kenichi Taniguchi
Dept. of Info. and Math. Science
Osaka University
Toyonaka Osaka 560-8531 JAPAN
{higashino, taniguchi}@ics.es.osaka-u.ac.jp

Abstract

In this paper, we propose a new bandwidth allocation technique where a new stream can preempt an appropriate amount of bandwidth from other existing streams, considering both quality and priority requirements of users. The existing preemption-based technique has focused on the preemption among the streams with the same user. However, in multi-party multimedia applications, there would be a case where we would like to raise the quality of a specific stream (e.g. in a video conference, the video stream of a new chair may become more important than those of the existing participants). Therefore the preemption should be allowed among streams with different users and the best preemption in terms of their requirements should be provided. In our technique, using the algorithm for solving the minimum flow cost problem, the preemption is calculated so that the total loss of quality and priority of the existing streams can be minimized. We have implemented the proposed technique and evaluated it through the experiment using MPEG1 video streams, and have confirmed that our technique can keep a high frame rate for each existing MPEG1 video stream even when accommodating a lot of new streams.

1. Introduction

Recent innovation of high speed networks has brought several new applications such as distributed

multimedia systems where continuous streams of video and audio are transmitted among multiple users in real time. In such applications, each stream should be delivered at a certain bit rate and each end system should have enough power for the satisfaction of users' requirements (e.g. color/BW and 30fps/15fps). For the purpose, a lot of techniques for providing quality of service (QoS) have been proposed for the efficient use of limited resources[1]. For the provision of QoS, the control mechanism for allocating an appropriate amount of network resources (e.g. bandwidth and minimum delay) and end system resources (e.g. processor power and memory) is required. Especially, several techniques for the bandwidth management have been proposed. Resource reservation techniques such as RSVP[2] enable the advanced reservation of a certain amount of bandwidth for each stream. The admission control techniques[5, 8] allow a new stream to be accommodated as long as it does not degrade the quality of existing streams. Resource sharing techniques[3, 4] allow several streams to share a certain amount of bandwidth.

Now we focus on bandwidth allocation techniques to allocate a proper amount of bandwidth to each stream, based on quality requirements and/or characteristics of traffics, in order to utilize the bandwidth efficiently. [3] has proposed a technique to allocate an amount of bandwidth to a group of streams so that they can share it. This technique is useful if there are a few (not so many) users to send streams at a time (e.g. in an audio conference, there are only several speakers among

a number of participants at a time). Although it can provide the efficient bandwidth allocation considering the characteristics of applications in some cases, it does not consider the quality requirements of users. [5] has proposed a technique considering users' requirements. It assumes that each stream is encoded by a layered-coding technique[9, 10] and a user who receives multiple streams can give a "quality value" to each layer of each stream. Based on the quality values, it is decided which layers/streams the user can receive. This technique allows a new stream with higher quality values to preempt the layers of existing streams with lower quality values. Therefore it can reflect the quality requirements flexibly. However, it restricts the preemption among the streams received by the same user. In general, in multi-party multimedia applications, the preemption should be allowed among streams received by different users (e.g. in a video conference, the video stream of a new chair may become more important than those of the existing participants, so a certain amount of bandwidth should be preempted from them) and the best allocation in terms of their requirements should be provided.

In this paper, we propose a bandwidth allocation technique where a new stream can preempt an appropriate amount of bandwidth from existing streams, based on quality and priority requirements of users. In this technique, for each stream, we assume that the followings are given by its user: (1) its minimum and maximum bandwidth requirements, (2) a quality value function representing its quality requirement and (3) a priority value representing its priority requirement. If there is no sufficient unused bandwidth for the request of a new stream, an amount of bandwidth can be preempted from the existing streams for the accommodation of the new stream, so that the total loss of quality and priority values can be minimized. The calculation can be done efficiently using an algorithm for solving the minimum flow cost problem[12]. Therefore the best preemption in terms of quality and priority among streams can be provided.

We have implemented our technique and evaluated it through the experiment using MPEG1 video streams. In the experiment, we have measured the change of frame rates of existing streams when accommodating new streams one by one. As a result, it is shown that our technique can keep a high frame rate for each MPEG1 video stream in existence of a lot of streams. Through our simulation, it is also shown that the preemption can be calculated in a reasonable time for several typical networks.

In Section 2, we will explain the proposed bandwidth allocation technique for unicast streams. In Section 3,

we apply our technique to multicast streams. The implementation technique is addressed in Section 4. The evaluation with some experiments is stated in Section 5. Section 6 concludes the paper.

2. Bandwidth Allocation for Unicast Streams

2.1. Admission Policy

A network is represented by an undirected graph. Each vertex and edge correspond to a computer node N_l ($1 \leq l \leq p$) (e.g. routers and hosts) and a communication link L_j ($1 \leq j \leq q$), respectively. The amount of unused bandwidth on L_j is denoted by $Bunused(L_j)$. A virtual path between two vertices without loops is called a (unicast) stream.

When the creation of a stream St_i is requested, the maximum and minimum bandwidth requirements (denoted by $Bmax(St_i)$ and $Bmin(St_i)$, respectively) must be declared. The currently allocated bandwidth of St_i is denoted by $Bcur(St_i)$. The sets of all the links and nodes used by St_i are denoted by $link_set(St_i)$ and $node_set(St_i)$, respectively.

Now suppose that there are $n - 1$ unicast streams St_1, \dots, St_{n-1} and the creation of a new unicast stream St_n is requested with $Bmax(St_n)$ and $Bmin(St_n)$. For the request, a routing protocol determines its path (i.e. $link_set(St_n)$ and $node_set(St_n)$ are determined). Then our protocol checks the following first admission condition.

[First Admission Condition]

$$\forall L_j \in link_set(St_n) [Bmin(St_n) \leq Bunused(L_j)]$$

If it holds, the request for St_n is accommodated since at least $Bmin(St_n)$ bandwidth can be allocated to St_n on every link in $link_set(St_n)$. If not, the preemption from other existing streams should be considered. Now for each stream St_k which uses some of links in $link_set(St_n)$, an amount of bandwidth $Bcur(St_k) - Bmin(St_k)$ is regarded as *preemptable*. Then for each $L_j \in link_set(St_n)$, the amount of $Bunused(L_j)$ and preemptable bandwidth on L_j is regarded as available and denoted by $Bunused_prep(L_j)$. Our protocol checks the following second admission condition.

[Second Admission Condition]

$$\forall L_j \in link_set(St_n) [Bmin(St_n) \leq Bunused_prep(L_j)]$$

If it holds, the new stream is accommodated with bandwidth preemption. If not, the request is refused.

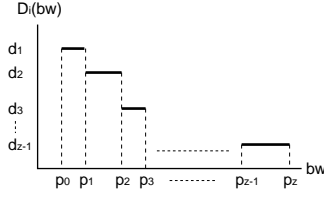


Figure 1. Quality Value Function

2.2. Requirements of Users

The requirements of users (or applications) for their streams are classified into two categories, quality and priority.

Quality Requirement The user of each stream separates preemptable bandwidth into several ranges and specifies a constant value for each range. Such a value is called a *quality value*, which represents how much quality is lost when a unit of the range is preempted. It can be represented as a function $D_i(bw)$ which returns a quality value per unit at currently allocated bandwidth bw , as follows.

$$D_i(bw) = \begin{cases} d_1 & (p_0 \leq bw < p_1) \\ d_2 & (p_1 \leq bw < p_2) \\ \dots & \dots \\ d_{z-1} & (p_{z-1} \leq bw \leq p_z) \end{cases}$$

Here, $p_0 = Bmin(St_i)$, $p_z = Bmax(St_i)$ and we assume $\forall s(0 \leq s \leq z-1) [d_{s-1} \geq d_s \geq 0]$. This is because video streams contain some of indispensable data which makes the lower bandwidth more important than the higher one. As a result, $D_i(bw)$ forms a down-stepping graph (Fig. 1).

For example, suppose that an MPEG1 video stream (320×240 pixels, 24 fps and 24 bit-colors) needs 0.3 Mbps, 0.5 Mbps and 0.7 Mbps for the transmission of I-, P- and B-frames, respectively. The user, who wants not so high but stable quality in the event of congestion, may define $D_i(bw)$ as follows.

$$D_i(bw) = \begin{cases} 10 & (0.3 \leq x < 0.8) \\ 1 & (0.8 \leq x \leq 1.5) \end{cases} \quad (1)$$

Another user, who wants quality according to the degree of congestion, may define it as follows.

$$D_i(bw) = 5 \quad (0.3 \leq bw \leq 1.5) \quad (2)$$

Here, for the fairness among streams, it is desirable that an average quality value per unit in $D_i(bw)$ is

almost the same as each other. For example, the one in the function (1) is $(10 \times (0.8 - 0.3) + 1 \times (1.5 - 0.8)) / (1.5 - 0.3) = 4.75$ and the one in the function (2) is obviously 5. The users who want high quality than others should use priority values explained below.

Priority Requirement The user of each stream St_i can also specify its priority as a positive value W_i (called a *priority value*). For example, if St_1 and St_2 have the priority values $W_1 = 1$ and $W_2 = 2$ respectively, the quality of St_2 needs to be kept higher than St_1 . Also, for the fairness, some rules to impose fees according to priority values and/or to negotiate those values among users may be desired to prevent every “greedy” user from requiring the highest priority at any time.

2.3. Problem Formulation

Assume that there are $n-1$ streams St_1, \dots, St_{n-1} on a network, with parameters $Bmin(St_i)$, $Bcur(St_i)$, $Bmax(St_i)$, $D_i(bw)$ and W_i for each stream St_i ($1 \leq i \leq n-1$). Also assume that the creation of a new stream St_n is requested with $Bmin(St_n)$ and $Bmax(St_n)$. According to our admission policy in Section 2.1, the first and second admission conditions are checked. If only the second admission condition holds, a certain amount of bandwidth (denoted by δ_i) is preempted from each St_i in order to allocate bandwidth $Bmin(St_n)$ to St_n . We treat the problem to decide δ_i ($1 \leq i \leq n-1$), where the total loss of *prioritized quality values* (explained later) is minimized.

The problem is formalized below.

[Restrictions]

$$\forall i(1 \leq i \leq n-1) [Bmin(St_i) \leq Bcur(St_i) - \delta_i] \quad (3)$$

$$\forall L_j \in link_set(St_n)$$

$$[Bmin(St_n) \leq Bunused(L_j) + \sum_{1 \leq k \leq n-1} f_{kj} \delta_k] \quad (4)$$

where

$$f_{kj} = \begin{cases} 1 & St_k \text{ uses } L_j (L_j \in link_set(St_k)) \\ 0 & \text{otherwise.} \end{cases}$$

[Objective Function]

$$\min \sum_{1 \leq i \leq n-1} \int_{Bcur(St_i) - \delta_i}^{Bcur(St_i)} Q_i(bw) dbw \quad (5)$$

Here, we define $Q_i(bw) = W_i D_i(bw)$ and call it a *prioritized quality value function*. $Q_i(bw)$ returns a *prioritized quality value* where both of quality and priority requirements are considered.

In the above expressions, the restriction (3) specifies the maximum amount of preemptable bandwidth from each St_i . The restriction (4) represents the second admission condition in Section 2.1. In the objective function (5), each term in the summation represents the loss of prioritized quality values when an amount of bandwidth δ_i is preempted from St_i . Therefore, the objective function (5) minimizes their total loss among all the streams.

2.4. Bandwidth Preemption Calculation

Each δ_i ($1 \leq i \leq n - 1$) which satisfies (3), (4) and (5) can be computed by the algorithm A below. Note that if St_n crosses another stream St_i on disjointed links, *i.e.*, the links in $link_set(St_n) \cap link_set(St_i)$ are not connected (see Fig. 5(a)), another algorithm B (explained later) is used.

Algorithm A An undirected graph is called a transportation graph if each edge e has a capacity $cap(e)$ and a cost $cost(e)$. The capacity $cap(e)$ means the maximum capacity of “transportation” through the edge e . The cost $cost(e)$ represents a cost per each unit of transportation. The algorithm first generates a transportation graph which represents the path of St_n as follows.

[Generation of Transportation Graph]

1. Generate the nodes which correspond to all the nodes in $node_set(St_n)$.
2. Add an edge “ L_j ” which represents each link L_j in $link_set(St_n)$, where $cap(L_j) = Bunused(L_j)$ and $cost(L_j) = 0$.
3. For each stream St_i ($1 \leq i \leq n - 1$) which uses at least one link in $link_set(St_n)$, if $link_set(St_n) \cap link_set(St_i)$ forms a connected path, add an edge “ St_i ” between both ends of the path. If not, use the algorithm B. Let $cap(St_i)$ and $cost(St_i)$ be $Bcur(St_i) - Bmin(St_i)$ (an amount of preemptable bandwidth) and $Q'_i(\delta_i)$, respectively. Here, $Q'_i(\delta_i)$ returns a prioritized quality value per a unit of preempted bandwidth δ_i , and is defined as $Q'_i(\delta_i) = Q_i(Bcur(St_i) - \delta_i)$. \square

For example, suppose that a stream St_1 uses links L_1 and L_2 , St_2 uses L_1 , L_2 and L_3 , and St_3 uses L_3 . Also suppose that the creation of a new stream St_4 on links L_1 , L_2 and L_3 is requested from a node s_4 to a node d_4 (see Fig. 2). We obtain the transportation graph shown in Fig. 3.

If we do not need to minimize the objective function (5), we only find “transportation paths” from s_4 to d_4 whose total capacity of transportation is $Bmin(St_4)$. For example, in Fig. 3, a path through the edges St_1 and L_3 with capacity 1 and a path through the edge St_2

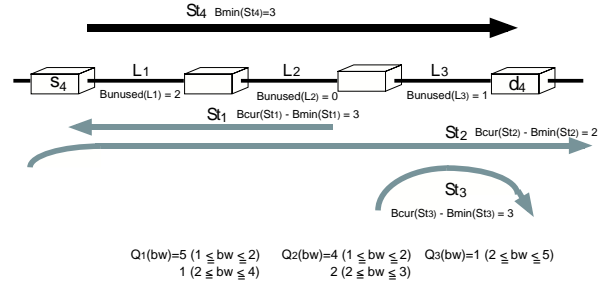


Figure 2. New Stream Request

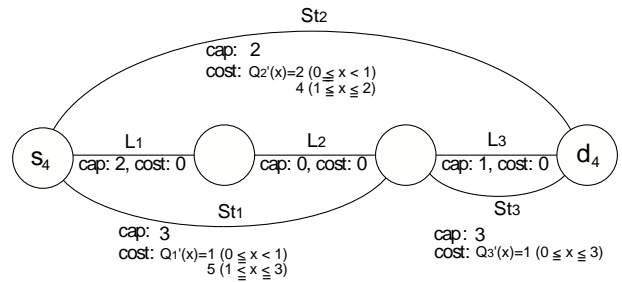


Figure 3. Transportation Graph

with capacity 2 are an example of such transportation paths whose total capacity is $Bmin(St_4) = 3$. It means that St_4 uses 1 unit of unused bandwidth on L_3 and preempts 1 and 2 units of bandwidth from St_1 and St_2 , respectively (that is, $\delta_1 = 1$ and $\delta_2 = 2$). Here, if we want to minimize the objective function (5) (which minimizes the total loss of prioritized quality values), we should find “transportation paths” from s_4 to d_4 with the minimum cost. Such a problem is known as the “minimum cost flow problem” and an algorithm has been investigated to decide transportation paths with the minimum cost within $O(m(m+n)(\log n)^2)$ (m and n denote the numbers of edges and nodes in the transportation graph, respectively)[12], if all the costs of edges in the transportation graph are constant values. In our case, the costs of edges may not be constant values because the quality value function is specified as a down-stepping graph as shown in Fig. 1. In such a case, the algorithm divides each edges into several ones.

[Path Division in Transportation Graph]

1. For each edge “ St_i ”, divide it into several ones so that each one has a constant cost. \square

For example, in Fig. 3, for a transportation path on St_1 with capacity 0.8, its cost is $0.8 * 1 = 0.8$. On

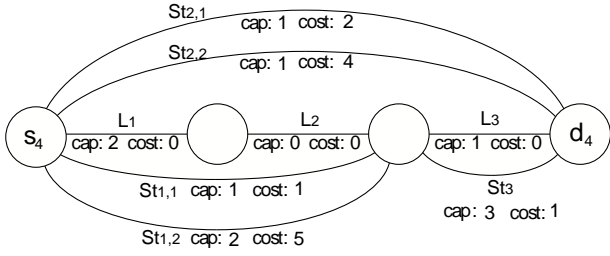


Figure 4. Path Division in Transportation Graph

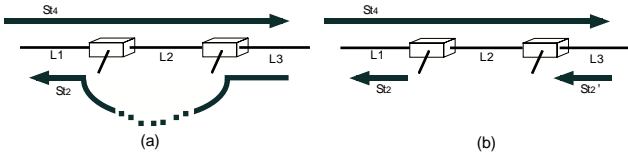


Figure 5. Streams Crossing on Disjoint Links

the other hand, for a path on St_1 with capacity 1.5, the cost is $1 * 1 + 0.5 * 5 = 3.5$. Therefore St_1 is divided into $St_{1,1}$ and $St_{1,2}$ whose costs are 1 and 5, respectively, and the modified graph in Fig. 4 is obtained (similarly St_2 is divided into two edges).

Now for the divided edges $St_{i,1}, \dots, St_{i,z}$, the algorithm must not select $St_{i,s}$ as a transportation path unless the capacity of $St_{i,s-1}$ is fulfilled as a transportation path. For example, if 1.5 units of bandwidth are preempted from St_1 , the algorithm must first find a path on $St_{1,1}$ with the full capacity 1 and then a path on $St_{1,2}$ with capacity 0.5. Here, from the definition of $Q_i(\delta_i)$, the cost of the edge $St_{1,2}$ is higher than that of $St_{1,1}$. Therefore, the algorithm always selects $St_{1,1}$ prior to $St_{1,2}$.

According to the above discussion, a set of transportation paths with the minimum cost can be obtained by using an algorithm to solve the minimum cost flow problem[12].

Algorithm B If $link_set(St_n) \cap link_set(St_i)$ does not form a connected path, the algorithm A cannot generate a transportation graph (see St_4 which crosses St_2 in Fig. 5(a)). In such a case, an algorithm to solve linear programming (LP) problems can be applied to the linear inequalities (3) and (4) and the objective function (5).

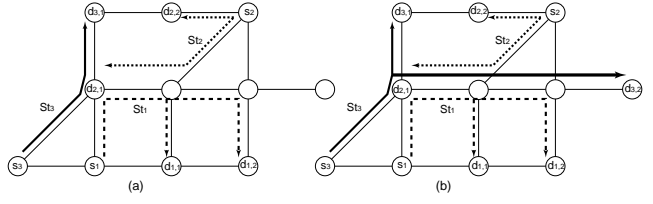


Figure 6. Connection Setup for Multicast Stream

The simplex method[12] is one of useful algorithms for LP problems, and in most cases it finds a solution within polynomial time. However, in the worst case it needs exponential time to find a solution, so it may not work efficiently if the size of a network is large. In such a case, we regard each connected path in $link_set(St_n) \cap link_set(St_i)$ as an independent stream and can get a semi-optimal solution by an ad-hoc way within polynomial time. For example, the stream St_2 in Fig 5(a) is separated into two streams St_2 and St_2' shown in Fig. 5(b).

2.5. Bandwidth Reallocation

Our protocol can also allocate an amount of unused (or released) bandwidth to the existing streams. The same algorithm can be used in order to maximize the total gain of prioritized quality values. Two major policies can be considered about when such allocation should be done: according to the request of users or in every pre-defined period.

3. Bandwidth Allocation for Multicast Streams

In multi-party multimedia applications, multicast communication is commonly used to prevent streams from wasting bandwidth. A multicast stream forms a tree structure where a source node and destination nodes correspond to its root and leaves, respectively. A node can join a multicast stream by connecting a unicast stream with the source node or one of the intermediate nodes of the multicast stream[13]. Therefore we can regard that a multicast stream with a source node s_n and destination nodes $d_{n,1}, \dots, d_{n,m}$ is constructed by connecting multiple unicast streams step by step as follows. First an initial stream $St_{n,1}$ from s_n to $d_{n,1}$ is generated, and then each branching stream $St_{n,k}$ is connected from an intermediate node of the current multicast stream to $d_{n,k}$ ($k = 2, 3, \dots, m$) repeatedly.

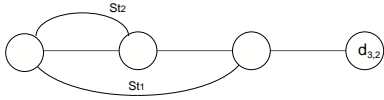


Figure 7. Transportation Graph

Fig. 6(a) and 6(b) show the generation of an initial stream $St_{3,1}$ from s_3 to $d_{3,1}$ and a branching stream $St_{3,2}$ from an intermediate node $d_{2,1}$ of $St_{3,1}$ to $d_{3,2}$, respectively. The initial and branching streams are regarded as unicast streams. Therefore the problem to generate a multicast stream can be treated as that to connect a new unicast stream $St_{n,k}$ over the existing multicast streams St_1, \dots, St_{n-1} and St_n where St_n has the initial and branching streams $St_{n,1}, \dots, St_{n,k-1}$.

Now we assume two cases. In the first case, all the destination nodes on a multicast tree St_i receive the stream St_i with the same bandwidth. We can easily see that the algorithms in Section 2.4 can be used for this case, since each multicast stream where an amount of bandwidth is preempted on a link releases the amount of bandwidth on all the links. For example, for the request of a new branching stream $St_{3,2}$ in Fig. 6(b), the algorithm forms a transportation graph shown in Fig. 7 (the costs and capacities are omitted).

In the second case, the destination nodes on a multicast tree St_i receive the stream St_i with different bandwidth from each other. This means that each intermediate node may drop some frames in a multicast stream according to the available amount of bandwidth on the link for each destination. Therefore the destinations with low bandwidth can also receive the stream. In this case, we should treat the multicast stream whose currently allocated bandwidth is different on each link. We have developed an algorithm to find an optimal solution by solving some LP problems and also an efficient algorithm to get a semi-optimal solution by solving only one LP problem for practical purposes. See [14] for the details.

4. Implementation

In this study, we have implemented our protocol as a bandwidth control system. The system consists of control servers and clients. Each server manages a specific domain of a network and each client manages streams on a receiver node.

Each server collects the information of the unused bandwidth and stream ID's on each link. It also keeps the information of the minimum/maximum bandwidth

requirements, currently allocated bandwidth and quality and priority values for each stream. When the server receives the creation request of a new stream from a client in its domain, it checks the two admission conditions described in Section 2.1. If bandwidth preemption is needed, the server lets each client release part of their bandwidth. Each client sends the creation request of a new stream when it receives a request from a user, and manages the bandwidth of the stream.

5. Evaluation

5.1. Experiment Using MPEG1

To evaluate the proposed method, we have implemented a pair of a video server and a client. A client receives and plays an MPEG1 video (320×240 pixels, 24 fps and 24 bit-colors) stored on the server over UDP/IP. Each server can drop some of B- and P-frames of the video while it is being transmitted to each client, according to an amount of bandwidth allocated by our protocol. Fig. 8 shows the network topology. The amount of bandwidth on each link is 10 Mbps and machines M_2, M_a, M_b and M_c are connected via a 10 Mbps switching hub. Machines M_1 and M_2 are connected directly. While two client machines M_a and M_b are receiving the streams St_a and St_b from the server M_1 , respectively, another client M_c asks either M_1 or M_2 to accommodate a new stream every 10 seconds. The quality value function $D_a(x)$ and $D_b(x)$ for St_a and St_b are given as (1) and (2) in Section 2.2. $D_a(x)$ means that P-frames are much important than B-frames. On the other hand, $D_b(x)$ means that B-frames and P-frames are equally important. We have given $W_a = W_b = 1$ as the priority values. Under the above condition, we have measured the change of the frame rates of MPEG1 video of St_a (from M_1 to M_a) and St_b (from M_1 to M_b) while the number of streams from M_1 or M_2 to M_c increases. M_c selects either $D_a(x)$ or $D_b(x)$ for each of its new stream at random.

For the comparison, we have used the three different algorithms in the experiment: (a) an algorithm to decide the allocation by priority, (b) a preemption algorithm which decides the preemption candidates by a heuristic way and (c) our algorithm. For the algorithm (a), we have assumed each new stream requires 1.5 Mbps bandwidth for its accommodation and St_b has a higher priority than St_a . The minimum required bandwidth is 0.3 Mbps for each stream and it will be disconnected if less than 0.3 Mbps bandwidth is available.

Fig. 9(a) and 9(b) show the change of the frame rates of St_a and St_b , respectively, when the number

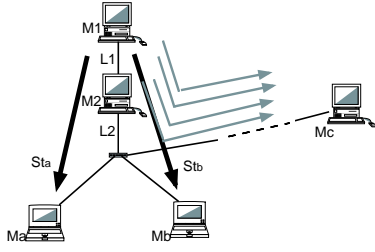


Figure 8. Experimental Environment

of new streams of M_c increases. In the algorithm (a), St_b could keep the high frame rate, while St_a stopped being played when M_c created the 9th stream. In the algorithm (b), both the frame rates of St_a and St_b went down earlier than our algorithm. Our algorithm could keep high frame rate for a longer time than the algorithm (b). Also, the number of new streams which could be accommodated in (a), (b) and (c) were 12, 18 and 19, respectively. These results show that our algorithm can realize high quality, accommodating a lot of streams.

5.2. Simulation Results

We have measured the following items via simulation where streams are created at random in the networks with several topologies shown in Fig. 10:

- (x). accommodation ratio of new streams,
- (y). average frame rates of streams,
- (z). average loss of frame rates of existing streams caused by preemption and
- (w). average bandwidth utilization.

When we calculate the average frame rates, we have included the streams which have not been accommodated, as the 0 fps streams. As the quality functions for the streams, we have used $D_b(x)$ in the previous section, and $D_c(x)$ and $D_d(x)$ defined below. Each new stream selects one of them at random.

$$D_c(x) = \begin{cases} 8 & (0.2 \leq x < 0.5) \\ 1 & (0.5 \leq x \leq 1.2) \end{cases}$$

$$D_d(x) = \begin{cases} 6 & (0.4 \leq x < 0.7) \\ 1 & (0.7 \leq x \leq 1.5) \end{cases}$$

Table 1 shows the simulation results. The algorithm (a) can accommodate many streams, but the frame rates are a bit low and are getting big effects by the preemption. The algorithm (b) can keep the high frame rates, but the accommodation ratio of new

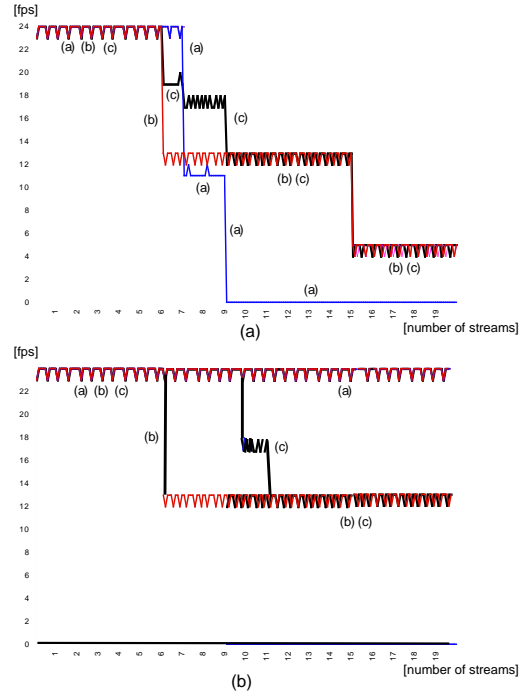


Figure 9. Change of Frame Rates

streams is relatively low. Our algorithm can keep high frame rates, accommodating a lot of streams.

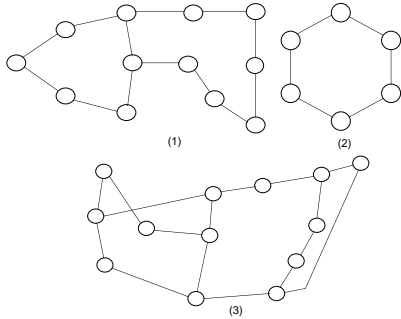
We have measured the calculation time of the algorithm A (on SONY NEWS 5000 with R4000 and 64 MB RAM). Even the number of preemption candidates is 30, the average calculation time is 11ms and is reasonably short. Also we have measured that of the algorithm B, using the simplex method. 99.87% of the calculation finished within 1000ms. When we used an approximation algorithm in Section 2.4, each calculation finished within 10ms. In this case, the maximum difference of total loss from the optimal solution was at most 10%.

6. Conclusion

In this paper, we have proposed a bandwidth allocation algorithm which allows a new stream to preempt a certain amount of bandwidth from existing streams, considering both quality and priority requirements. We have also implemented our algorithm and experimented with MPEG1 video streams for the evaluation. In our algorithm, an appropriate amount of bandwidth is preempted by a new stream so that the total loss of quality and priority in the existing streams can be minimized. The experimental results show that our algorithm can

Table 1. Simulation Results on Various Network Topologies

param.	[unit]	topology(1)			topology(2)			topology(3)			average		
		(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
(x)	[%]	81.7	66.7	80.0	47.0	48.0	49.0	49.0	46.7	51.0	59.0	53.8	60.0
(y)	[fps]	13.1	16.7	17.0	10.0	7.4	10.1	12.1	13.9	16.1	11.7	12.7	14.4
(z)	[fps]	5.6	4.6	2.6	10.0	7.5	4.9	5.6	6.7	2.9	7.1	6.3	3.5
(w)	[%]	69.6	70.0	76.4	77.0	80.5	84.4	63.0	61.1	66.1	69.9	70.5	75.6

**Figure 10. Network Topologies**

accommodate more streams with the satisfaction of existing streams' requirements than other algorithms.

We have used MPEG1 coding scheme in the experiment. We are going to apply our algorithm to other coding schemes popular in video conference systems like H.261[11]. Until now, several QoS routing methods[7] have been proposed to find a preferable route matching given user requirements. These techniques can be used to find a route with the largest unused bandwidth. Our technique can be applied to such a route for the efficient bandwidth utilization.

References

- [1] C. Aurrecochea, A. Campbell and L. Hauw, "A Survey of QoS Architectures," *Multimedia Systems Journal*, Special Issue on QoS Architecture, 1998. (to appear)
- [2] P.P. White, "RSVP and Integrated Services in the Internet : A Tutorial," *IEEE Communications Magazine*, Vol. 35, No.5, pp.100-106, 1997.
- [3] A. Gupta, W. Howe, M. Moran and Q. Nguyen, "Resource Sharing for Multi-party Real-time Communication," *Proc. of INFOCOM'95*, pp.1230-1237, 1995.
- [4] L. Libman and A. Orda, "Atomic Resource Sharing in Noncooperative Networks," *Proc. of INFOCOM'97*, pp.1008-1015, 1997.
- [5] N. Shacham, "Preemption-Based Admission Control in Multimedia Multiparty Communications," *Proc. of INFOCOM'95*, pp.827-834, 1995.
- [6] R.T. Apteker, J.A. Fisher, V.S. Kisimov and H. Neishlos, "Video Acceptability and Frame Rate," *IEEE Multimedia*, Vol. 8, No. 10, pp.32-40, 1995.
- [7] W.C. Lee, M.G. Hluchyi and P.A. Humblet, "Routing Subject to Quality of Service Constraints in Integrated Communication Networks," *IEEE Network*, Vol. 9, No. 4, pp.46-55, 1995.
- [8] S. Jamin, S.J. Shenker and P.B. Danzig, "Comparison of Measurement-based Admission Control Algorithm for Controlled-Load Service," *Proc. of INFOCOM'97*, pp. 975-982, 1997.
- [9] 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.
- [10] N. Shacham, "Multipoint Communication by Hierarchically Encoded Data," *Proc. of INFOCOM'92*, pp.2107-2114, 1992.
- [11] "Video Codec for Audiovisual Services at $p \times 64$ kbits," *ITU-T Recommendation H.261*.
- [12] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, "Network Flows: Theory, Algorithms and Applications," Prentice-Hall, 1993.
- [13] 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.
- [14] H. Sakate, H. Yamaguchi, K. Yasumoto, T. Higashino and K. Taniguchi, "A Bandwidth Preemption Technique in Multicast Communication," *ICS Research Report*, ICS-98-1, 1998.