Vol. 44 No. 2 IPSJ Journal Feb. 2003

Recommended Paper

Multicast Techniques for Personalized Media Stream Delivery

Katsuhiko Sato[†,††] and Michiaki Katsumoto[†††]

This paper describes multicast techniques for a network distribution method designed to accommodate an innovative broadcast content delivery service that provides temporally and spatially personalized video/audio contents. Although multicasting is unsuitable for on-demand distribution and heterogeneous media distribution via real-time streaming transfer, we are attempting to use it to achieve effective network resource utilization. As far as we are aware, this has not been previously attempted. We consider two multicasting techniques — asynchronous and layered multicasting — and describe how we have improved them for this purpose, and developed two new theoretical methods. One is a statistical traffic-control algorithm based on asynchronous multicasting that enables flexible bandwidth control to meet the network bandwidth design requirement while reducing the use of a receiver's buffer memory. The other is a multicast technique for the delivery of diverse media segments/objects, which is based on a layered multicast technique that manages the fewest possible multicast groups through cooperative asynchronous multicasting. To demonstrate the implementability of these two methods, we describe a network design that uses the latest Internet technologies. We also discuss the effectiveness of this scheme for traffic reduction on the basis of a numerical analysis and simulation results.

1. Introduction

A number of Internet services have been developed that provide a form of broadcasting called webcasting[1)], which may become the basis for a new form of Internet application. webcasting services provide live broadcasting and on-demand content delivery. They are usually used with real-time streaming transfer and are resorted to in some multicast or decentralized cache techniques to reduce the load on the network. Meanwhile, the broadband infrastructure for content distribution based on Internet technologies has grown rapidly and the quality of video/audio delivery may soon become equal to or exceed that of current TV broadcasting[2)].

We have been trying to create new applications and techniques on the premise that the broadband infrastructure for content distribution will become a reality. For innovative broadcast content delivery over the next-generation broadband Internet, we have proposed Personalized Media Stream Delivery (PMSD)[3)], which is a system characterized by the following features:

- A wide range of video/audio content based on specific information regarding each user can be produced. That is, several kinds of video segments (scenes) and objects prepared in the server system are compiled on the basis of the user's personality (e.g., the user's interests or lifestyle).
- Through the broadcast content delivery service, such customized content can be delivered at the time of the user's choosing by streaming with an optimized level of quality based on the computational ability of the receiver system to play back the content.

This paper focuses on a scheme of effective distribution for the PMSD. The distribution of temporally and spatially personalized video/audio contents consumes a considerable amount of network resources. Here, we consider two types of multicasting, asynchronous multicasting and layered multicasting, and propose two new techniques. One is a statistical traffic-control algorithm that improves asynchronous multicasting, which enables effective distribution of temporally personalized content delivery. The other is a technique for the delivery of diverse media segments/objects that improves layered multicasting, which enables ef-

† Japan Radio Co., Ltd, Laboratory, Network Research Group
†† The University of Electro-Communications, Graduate School of Information Systems
††† Communications Research Laboratory, High-Speed Network Section

The initial version of this paper was presented at the DPS workshop held on Oct. 2001, which was sponsored by SIGDPS. This paper was recommended to be submitted to the Journal of IPSJ by the chairperson of SIGDPS.

fective distribution for spatially personalized content delivery. We demonstrate the implementability of the two techniques by designing network models and protocols using the latest Internet technologies, and show the effectiveness of traffic retrenching through a numerical analysis. We evaluate the statistical traffic-control algorithm with regard to our simulation results.

2. Asynchronous Multicasting and Layered Multicasting

The easiest way to enable distribution of temporally and spatially personalized content, namely, "on-demand content distribution" and "heterogeneous media content distribution", is unicasting, in which content items are carried to each user in separate flows. However, this method uses up a lot of network resources. With a multicast method, the contents are carried to each user in a single flow that is copied at appropriate network nodes depending on the location of the receiving user. Hence, the use of network resources can be significantly reduced. Unfortunately, multicasting is essentially inapplicable to on-demand distribution and heterogeneous media distribution via real-time streaming transfer, since the same content must be delivered at the same time to the same multicast group members.

2.1 Asynchronous Multicasting

Asynchronous multicasting has been studied for on-demand content delivery. The principle of asynchronous multicasting is that a portion of overlapping data in the delivery of any particular content is aggregated into a multicast flow sent to users whose requests are made at about the same time, and the data not included in the multicast flow (i.e., the portion between the time when the multicast flow starts and the time when a request occurs) are individually delivered by unicast flows. The data arriving at the receiver system, which are aggregated into a multicast flow, are not immediately played back, but are buffered until the data delivered by a unicast flow has been completely played back.

Several methods have been studied for asynchronous multicasting. A fast-transfer method is described in Woo and Kim[4] and Kalva and Fuhrt[5], where the content is divided into small data units and the data-transmission rate is three to four times as fast as the rate at which the data are played back. A stream-transfer method has been proposed in Carter and Long[6], where the content is transmitted at the same rate as it is played back. In Uno and Tode, et al.[7], the portion of data delivered by a unicast flow is transmitted in a burst manner.

2.2 Layered Multicasting

Layered multicasting has been studied for delivering content with different levels of quality determined according to the user's processing ability. This technique is premised on the use of multi-layered video coding techniques. Each piece of layered encoding data is mapped onto a different multicast group, and the receiver system joins multiple multicast groups according to its computational ability. There have been a number of published reports concerning layered multicasting. For example, an experimental implementation to develop a standard is reported in Suzuki and Mimura, et al.[9], and McCanne et al. have proposed that receiver systems be able to dynamically change to obtain the number of multicast groups depending on the degree of network congestion[8].

3. Multicast Techniques Applied for PMSD

3.1 Improving Asynchronous Multicasting for Time-Personalized Media Distribution

The work reported in Refs. 4), 5), 6), and 7) was aimed at reducing the load on the server system. In other words, the objective was to minimize the traffic. To achieve effective distribution, we must consider ways to fully utilize a given network resource in addition to minimizing the traffic. Moreover, asynchronous multicasting essentially consumes buffer memory at the receiver system, so we must consider ways to reduce the use of the receiver's buffer memory. Hereafter, we focus on ways to improve conventional asynchronous multicasting on a stream basis, as in Ref. 6).

We consider the relationship between the cycle of aggregating flows into a multicast and the arrival rate of requests for delivery (and the length of the content), and express the traffic intensity of stream-based asynchronous multicasting as a numerical formula. We statistically control the traffic by controlling the generation rate of the multicast flow on the basis of the request rate. Taking into consideration of the trade-off between the traffic intensity and the use of the receivers' buffer memory, we can re-

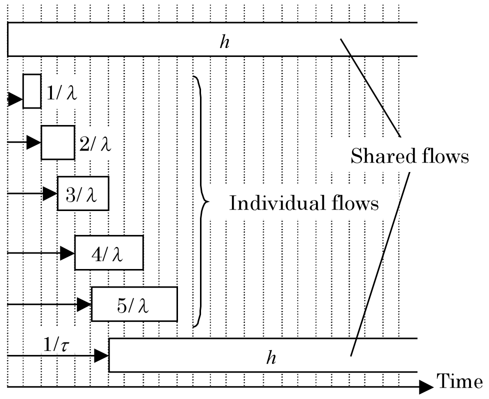

Fig. 1 Asynchronous multicasting.

duce buffer memory use by making full use of the available network bandwidth. Hence, we propose an algorithm for statistical traffic control that enables flexible bandwidth adjustment while striving for overall reduction of the use of the receivers' buffer memory. If we assign the available bandwidth for each content delivery according to each request rate and adjust the traffic to the available bandwidth, we can further reduce the use of the receivers' buffer memory and efficiently utilize network resources.

3.2 Statistical Traffic-Control Algorithm

Let us consider the traffic intensity in stream-based asynchronous multicasting. We assume that the provided content is transmitted at a constant bit rate, that its length is h, and that delivery requests occur at random (i.e., the average arrival rate of requests is λ, and the average arrival interval is $1/\lambda$). **Figure 1** shows that one delivery to a receiver is made by multicasting as a shared flow, and subsequent deliveries to other receivers are made by unicasting as individual flows.

There are two ways to determine the length of individual flows: one is to make the length of every individual flow the same—equal to the generation interval of the shared flows; the other is to make the length of each individual flow correspond to the period of time between the start of a shared flow and the start of an individual flow. (Fig. 1 shows the latter.) Although the former can be implemented more easily, the latter can reduce traffic more effectively.

The length of an individual flow may be expressed as the number of segments (or subdivided segments), which is the same as the unit used for the delivery of diverse segments. In

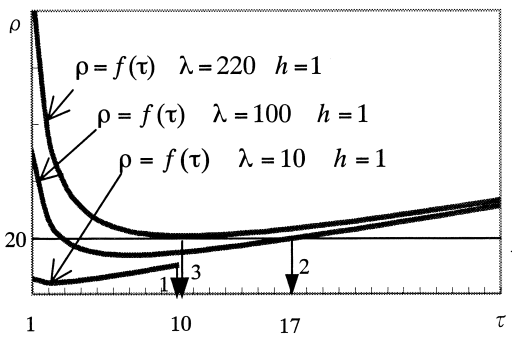

Fig. 2 Control of the traffic intensity.

this way, the server and receiver systems do not require highly accurate time processing to determine the length of individual flows.

Here, the generation rate of shared flows is expressed as τ ($\tau \leq \lambda$). When the length of every individual flow is the same, it is equal to the generation interval of the shared flows (i.e., $1/\tau$), and thus the traffic intensity, ρ, is

$$\rho = \tau h + (\lambda - \tau)\frac{1}{\tau} = \tau h + \frac{\lambda}{\tau} - 1 \quad \text{[erl]} \tag{1}$$

When the length of each individual flow is different, the number of individual flows between two shared flows is $\lambda/\tau - 1$, the length of each individual flow is $1/\lambda, 2/\lambda, 3/\lambda \cdots$, and the average length is then $1/2\tau$. Hence, the traffic intensity is

$$\rho = \tau h + (\lambda - \tau)\frac{1}{2\tau} = \tau h + \frac{\lambda}{2\tau} - \frac{1}{2} \quad \text{[erl]} \tag{2}$$

In both equations, $\rho = f(\tau)$ is a downward convex curve, and ρ takes the minimum value when $\tau = \sqrt{\lambda/h}$ in Eq. (1), or $\tau = \sqrt{\lambda/2h}$ in Eq. (2). Hence, it is possible to maintain the traffic intensity at a minimum all the time by updating τ to the most appropriate value according to the observed λ and h.

The average use of buffer memory at the receiver (the buffer is used for the data of the shared flow and its size corresponds to the length of the individual flow) is reduced as τ increases. Therefore, in determining τ, there is a trade-off between minimizing the traffic intensity and reducing the use of buffer memory.

Provided that the upper bound of the traffic intensity (i.e., the available bandwidth) is given as A, τ is determined as follows (**Fig. 2**). Hereafter, we describe the case where the length of each individual flow is different.

When the observed $\lambda \times h$ is below A, the traffic intensity does not exceed the upper bound even if all contents are delivered by unicasting. Therefore, τ is set to λ (i.e., all contents are delivered as a shared flow), so that the use of buffer memory at the receiver will be zero (arrow 1 in the figure).

When the observed $\lambda \times h$ is greater than A and the minimum value of ρ (i.e., $f(\sqrt{\lambda/2h}) = \sqrt{2\lambda h} - 1/2$) is smaller than A, τ is set to the largest value satisfying $f(\tau) \leq A$. Here, the equation is

$$A = \tau h + \frac{\lambda}{2\tau} - \frac{1}{2} \quad \text{[erl]} \tag{3}$$

and the solutions for τ are

$$\tau = \frac{1/2 + A \pm \sqrt{(1/2 + A)^2 - 2\lambda h}}{2h} \tag{4}$$

The larger one is selected as τ (arrow 2 in the figure).

When A is smaller than the minimum value of ρ (i.e., $f(\sqrt{\lambda/2h}) = \sqrt{2\lambda h} - 1/2$), τ is set to $\sqrt{\lambda/2h}$ to minimize the traffic intensity (arrow 3 in the figure).

Thus, we enable flexible traffic adjustment while striving for overall reduction of the use of a receiver' s buffer memory. In addition, we can ensure maximum use of buffer memory (i.e., $1/\tau$) by determining the minimum value of τ. If a server receives a request from a receiver whose buffer memory is less than $1/\tau$, it has only to generate a new shared flow.

3.3 Improving Layered Multicasting for Spatially Personalized Media Distribution

The essence of layered multicasting is that the common part of data to be received by users is aggregated into a multicast. Conventional layered multicasting delivers content with different levels of quality in multi-layered coding video, where the data are aggregated with respect to quality (i.e., each piece of layered encoded data is aggregated into a multicast as common data). Here, we extend this idea to the delivery of diverse video segments and objects; that is, we try to aggregate data with respect to the segment and object. A segment here means a unit that constitutes a video program, and an object means a unit that constitutes a video picture (e.g., background objects and foreground moving objects).

In addition, we consider here a method for mapping a multicast group onto each aggregated segment or object, and show how the number of multicast groups can be reduced to alleviate the burden of the multicast handling task on the network.

3.4 Technique for Delivery of Diverse Segments and Objects

For the delivery of diverse segments, the segments that are common among users are aggregated, and each is mapped onto a different multicast group. The receiver systems dynamically change to join the multicast groups and play back the segments of their own choosing.

For the delivery of diverse objects, all objects that every user may possibly need are contained in one segment mapped onto one multicast group. Objects to be played back are chosen from among all objects in the segment according to the user's needs. This method does not require a complicated implementation such as grafting and pruning of multicast trees for each object.

In the delivery of diverse segments, a number of multicast groups (and addresses) are used for mapping onto each segment, which forces the network to frequently construct the trees for each multicast group. There is a substantial burden on the network tasks that are used to handle multicast management, so we need to consider ways to reduce the number of multicast groups to be used.

The server system produces several patterns of a series of segments and relates them to user groups that represent typical users. The server determines which user group the user requesting content delivery belongs to. For the user who requests first, the server system maps the same multicast group onto all of the series of segments to be delivered. For users who make a request after that, the server system maps a new multicast group onto only the segments that are different from those that have already been requested for delivery to other users.

Even though this method can reduce the number of multicast groups, a problem arises. In stream-based transmission, each segment is transmitted at the same rate as it is played back, and a series of segments are played back continuously. Therefore, the transition of a multicast group must be performed instantly. However, there is overhead time during which the network constructs a multicast tree, and the server cannot transmit the segment until the corresponding multicast tree is completely formed. In the above method for multicast group assignment, the same multicast group is

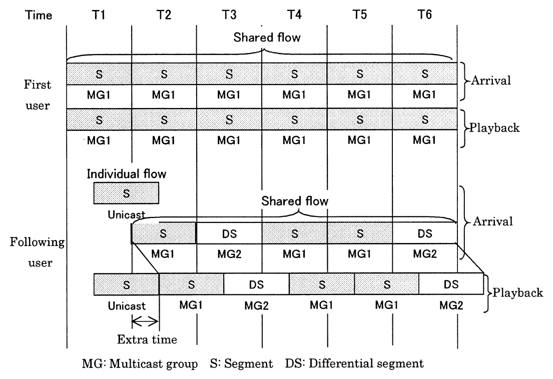

Fig. 3 Multicast group assignment for the delivery of diverse segments.

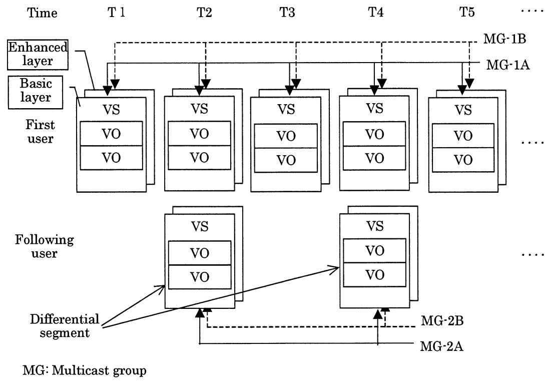

Fig. 4 Delivery of diverse segments and objects using MPEG-4.

used for one or more segments, and thus the multicast tree of a certain segment cannot be built during the distribution of another segment to which the same multicast group was assigned; that is, building of a multicast tree prior to transmission of a segment is not allowed.

We can solve this problem by using asynchronous multicasting (**Fig. 3**). For the first user, who will receive only the shared flow in asynchronous multicasting, the transition of multicast groups is not allowed while the content is being delivered, because the user plays back the received segment as soon as it arrives at the receiver system. For the following users who will receive both individual and shared flows, the transition of multicast groups is allowed, because there is extra time between the arrival of the segment at the receiver and its playback (i.e., segments in the shared flow are buffered until all segments in the individual flow are completely played back), which enables the network to build multicast trees.

Figure 4 shows the delivery of diverse qualities, segments, and objects by means of MPEG-4, which is based on the above method. MPEG-4 video consists of video object sequences (VSs) that correspond to segments here. A VS consists of video objects (VOs). A VO consists of video object layers (VOLs), which are layers of temporal-spatial resolution. A VOL consists of video object planes (VOPs). Note that here the layers of resolution are determined per VS rather than per VO.

4. Considerations for Implementation

To demonstrate the implementability of the above two methods, we will describe a network design that uses the latest Internet technologies as an example of how the methods can be used.

4.1 Requirements

A major premise regarding the delivery of video/audio content is that the network must provide a consistent QOS while streaming all of the content data. Policies like those for best-effort forwarding in the network and quality manipulation at the receiver system are not used here. This is because we assume that content delivery is charged for or accompanied by advertisements, as in current broadcast TV services.

4.2 Basic Concept of Network Control

Generally, there are two network models for network control: centralized control and distributed control. In the former, one specific control system maintains the only database and executes routing and signaling functions to control the whole network. In the latter, the individual network nodes autonomously exchange information about topology and resources, maintain the same databases, and execute routing and signaling functions collaboratively.

To guarantee the QOS of streaming delivery, let us consider two strategies: a provisioning-based QOS guarantee and a scrambling-based QOS guarantee. The former guarantees the QOS by estimating the statistical traffic for multiple flows and over-investing network resources. The latter guarantees the QOS through signaling (e.g., for admission control and resource reservation) for each delivery before sending data. As a result, users scramble for the limited available network resources.

4.3 Network Design

We described some models and design considerations in a previous publication [3]. We believe that centralized network control would

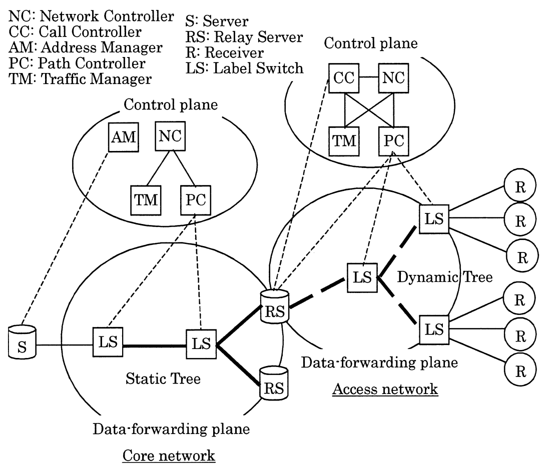

Fig. 5 Network model.

be easier at an early stage of implementation. In the core network, where the traffic is concentrated, a multicast tree is statically constructed and a provisioning-based QOS guarantee is used. In the access network where there are limited resources, multicast trees are dynamically constructed and a scrambling-based QOS guarantee is used. In addition, we use MPLS (Multi-Protocol Label Switch)[11] technology in both networks, which provides traffic engineering to meet the QOS requirements by establishing connections called LSPs (Label Switch Paths) for each FEC (Forwarding Equivalence Class).

Figure 5 shows our network model, which consists of a single core network and multiple access networks. Both the core and access networks are physically decoupled into a data-forwarding plane and a control plane. RSs (Relay Server systems) are deployed at the border between the networks. The S (Server system) initiates shared flows only, and the RSs initiate individual flows and flows for differential segments (for the delivery of diverse segments) and relay shared flows. Thus, the traffic in the core network can be further reduced.

Hereafter, we present an outline of our network design. For details of the protocol design, please refer to appendix A.

4.3.1 Core Network

The data-forwarding plane consists of LSs (Label Switch systems) that can execute Diffserv (Differentiated Services)[10]. Diffserv provides a provisioning-based QOS guarantee within a closed network (a DS domain) by estimating the traffic for multiple flows as a unit of service. A static multicast tree is constructed by setting up a permanent LSP onto which every shared flow sent from the S is carried. EF-PHB (Expedited Forwarding-PHB)[13] is applied to the LSP as the highest class of forwarding service. Every LS in a DS domain must guarantee the minimum outgoing rate for such a class, and an ingress LS in a DS domain must constrain incoming traffic not to exceed the minimum outgoing rate.

The statistical traffic control algorithm (described in Section 3.2) is useful as a means of constraining incoming traffic. The S can control traffic by considering the minimum outgoing rate of EF-PHB as the upper bound of traffic "A."

In the control plane, an NC (Network Controller system) controls the network utilization policy. In particular, it constructs a static multicast tree and determines the allocation of network resources. The TM (Traffic Manager system) maintains a topology database with QOS parameters. The AM (Address Manager system) manages multicast addresses within a local scope. The S obtains multicast addresses from the AM and forwards them to the R (Receiver system) before transmitting content data. The PC (Path Controller system) distributes label information to the LSs along the multicast tree to establish the LSP.

4.3.2 Access Network

An FEC is assigned for each micro-flow, be it a shared flow, an individual flow, or a flow for the differential segments (for the delivery of diverse segments), and an LSP is dynamically set up by means of control-plane signals whenever a new micro-flow is initiated. To ensure the QOS, every LS monitors the utility of its links (available bandwidth) and performs policing and shaping for each micro-flow. The CC (Call Control system) is deployed in the control plane to process signaling functions that establish and release the LSPs within the access network.

For the delivery of diverse segments, the RS does not send a differential segment until a corresponding multicast tree is completely constructed. Therefore, the CC coordinates the RS sending differential segments and the PC setting up LSPs.

To consistently guarantee the QOS for the delivery of the full set of content data, the bandwidth of the links along the tree for all multicast groups that one R is supposed to join must be secured beforehand. Therefore, the TM maintains a resource-management database on a time axis, and admission control and resource reservation are done over a span of time rather than at a point of time.

4.4 Numerical Analysis

We calculated the traffic intensity in the core network and the access networks, where an algorithm for statistical traffic control was implemented, and evaluated our network design. We measured the effectiveness of retrenching the traffic intensity at a trunk link in the access networks and at the trunk and branch links in the core network by comparing the traffic intensity in our method with that for unicast delivery. (Note that here our algorithm worked to minimize the traffic and we do not discuss the use of the receiver's buffer memory). Although we stated that the length of individual flows may be expressed as the number of segments (or sub-segments), we assume that a segment is very small and that the length of individual flows can be flexibly changed. The number of access networks attached to the core network is expressed as m and the average rate of requests for delivery in one access-network is expressed as λ.

In the access network, both shared flows and individual flows are transmitted. When $\tau = \sqrt{\lambda/2h}$, the traffic intensity is lowest. Substituting this into Eq. (2), we obtain the traffic intensity as

$$\rho = \sqrt{2\lambda h} - 1/2 \quad \text{[erl]} \tag{5}$$

In unicast delivery, the traffic intensity at the trunk link in the access network is $\lambda \times h$. Therefore, the ratio of traffic intensity in our multicast technique to that in unicasting, R_{at}, is

$$R_{at} = \frac{\sqrt{2\lambda h} - 1/2}{\lambda h} \tag{6}$$

In the core network, only the shared flows are transmitted, and the traffic intensity at trunk and branch links is the same. When $\tau = \sqrt{\lambda/2h}$, the traffic intensity is lowest, so the traffic intensity is

$$\rho = m\tau h = m\sqrt{\frac{\lambda h}{2}} \quad \text{[erl]} \tag{7}$$

In unicast delivery, the traffic intensity at the trunk link is $m \times \lambda \times h$, and the traffic intensity at a branch link is $\lambda \times h$. Therefore, the ratios of traffic intensity in our multicast technique to those in unicasting at the trunk link, R_{ct}, and

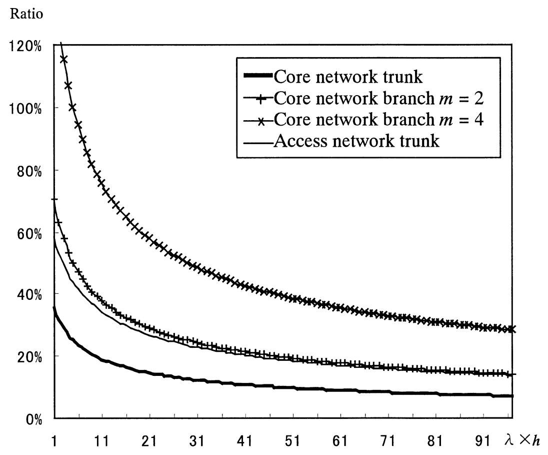

Fig. 6 Ratio of multicast traffic intensity with the proposed technique to unicast traffic intensity.

at a branch link, R_{cb}, are

$$R_{ct} = \frac{m\sqrt{\lambda h/2}}{m\lambda h} = \frac{1}{\sqrt{2\lambda h}} \tag{8}$$

$$R_{cb} = \frac{m\sqrt{\lambda h/2}}{\lambda h} = \frac{m}{\sqrt{2\lambda h}} \tag{9}$$

Figure 6 shows the ratio of traffic intensity in our multicast technique to that in unicasting in the core network and the access network. The effect of retrenching the traffic increases as the request rate rises. When $\lambda \times h$ is 100, the ratio is only 7% at the trunk link in the core network and 13% in the access network. The effect of traffic retrenching decreased, though, as m is increased. When m is 4 and there are few requests ($\lambda \times h$ is below 8), the ratio is over 100% at the branch link in the core network. We must therefore carefully design the number of branches on the basis of the estimated number of requests.

5. Simulations

We evaluated the operation of the proposed statistical traffic-control algorithm through a simulation. The simulation network model was identical to the above access network. The arrival rate of requests, λ, had a Poisson distribution for a given average value. This average value varied with time, so the request rate fluctuated periodically according to the time of day. We assumed that the average rate of requests started at 5 requests/hour (the sparsest), increased to 30 in 12 hours, and then decreased to 5, again in 12 hours. The content length was 2 hours.

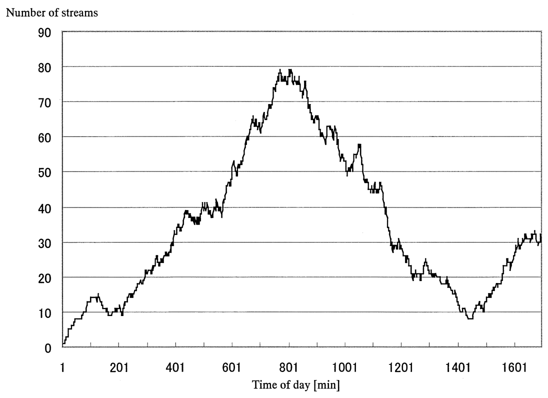

Fig. 7 Traced traffic (number of streams) in unicasting.

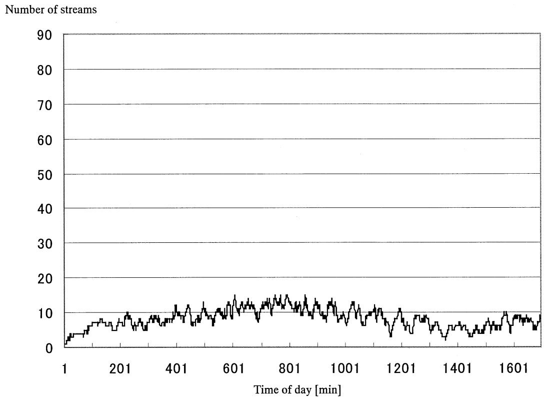

Fig. 8 Traced traffic (number of streams) in our multicasting technique, where τ was set to the value that minimized traffic.

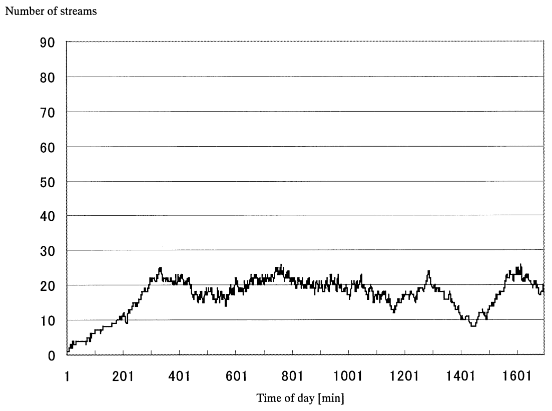

Fig. 9 Traced traffic (number of streams) in our multicasting technique, where τ was set to the value that allowed the traffic to reach 20.

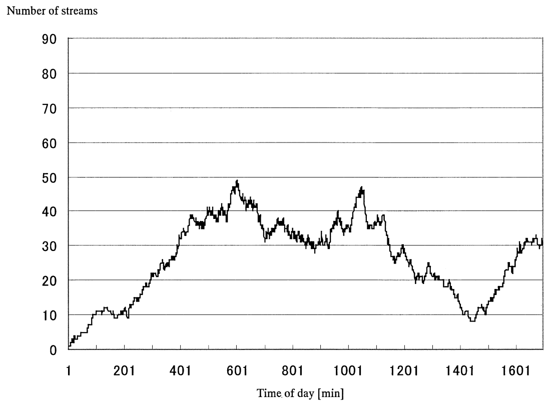

Fig. 10 Traced traffic (number of streams) in our multicasting technique, where τ was set to the value that allowed the traffic to reach 40.

5.1 Simulation results

Figures 7, **8**, **9**, and **10** show the traffic intensity (the number of streams) on the trunk link in the access network that resulted from the simulation. Figure 7 shows the results obtained for a current unicast method, and Figs. 8, 9, and 10 show the results obtained for a multicast method with the proposed traffic-control algorithm. We set τ to the value that minimized the traffic at all times (Fig. 8) and to the value that allowed the traffic to be as high as 20 or 40 (Figs. 9 and 10, respectively) to reduce the use of buffer memory at the receiver system.

In Fig. 7, the number of streams reached about 80 at the 12th hour, while the number of streams in Fig. 8 remained at about 15 at the 12th hour. The ratio is identical to the result calculated by using Eq. (6).

The traffic in Fig. 9 was traced in the same way as in Fig. 7 until the number of streams exceeded 20 (at about the 300th minute). Here, τ was set to λ, so all contents were delivered in a shared flow. That is, the traffic was practically the same as that in the unicast method. After about the 300th minute, the traced traffic fluctuated around 20. τ was set to the largest value that would allow the traffic to reach 20 according to Eq. (4). After about the 1300th minute, the same traffic was again traced as in Fig. 7. τ was set to λ, so all contents were delivered in a shared flow. That is, the traffic was practically the same as in the unicast method.

The situation for the Fig. 10 results was similar to that for Fig. 9. When the traffic was under 40, τ was set to λ, so that the traffic was traced in the same way as in Fig. 7. Otherwise, τ was set to the largest value that allowed the traffic to reach 40.

Figures 11, **12**, and **13** show the recorded use of buffer memory at the receiver systems. The horizontal axis shows the number of con-

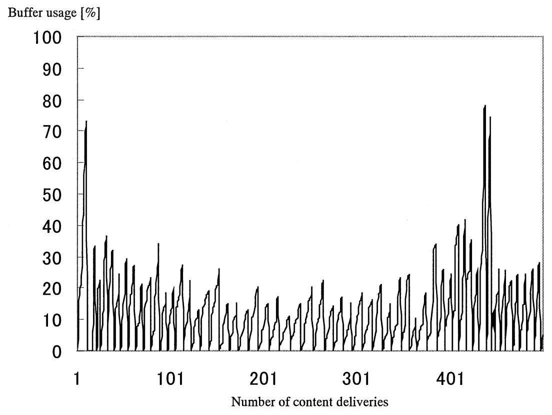

Fig. 11 Recorded use of buffer memory in receiver systems (τ was set to the value that minimized traffic).

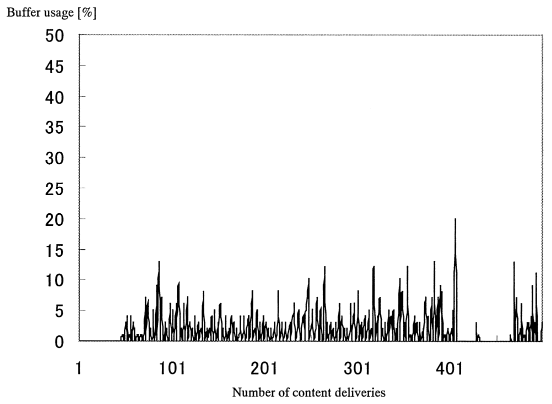

Fig. 12 Recorded use of buffer memory in receiver systems (τ was set to the value that allowed the traffic to reach 20).

tent deliveries (requests for content delivery were generated 500 times in this simulation), and the vertical axis shows the use of buffer memory expressed as a percentage of the content length. In Fig. 11, τ was set to the value that minimized traffic at all times. Use of the buffer memory reached about 80% of content at worst. In Figs. 12 and 13, τ was set to the value that allowed the traffic to reach 20 and 40, respectively. The use of buffer memory was at most about 20% and 10%, respectively.

5.2 Remarks

Our simulation results indicated the following:

First, the traced traffic in Figs. 7 and 8 indicates that our statistical traffic-control algorithm can reduce traffic significantly compared with the quantity in unicasting.

Second, the traced traffic in Figs. 9 and 10 confirms the validity of the numerical calculation derived in this paper. We can adjust the traffic at will by giving the available bandwidth and determining the generation rate of the shared flows on the basis of the equations given in section 3.

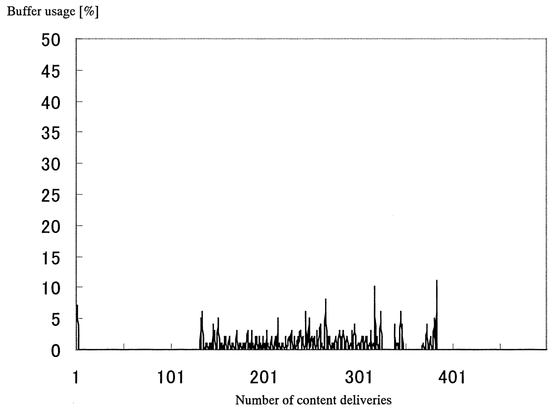

Fig. 13 Recorded use of buffer memory in receiver systems (τ was set to the value that allowed the traffic to reach 40).

Third, Figs. 11–13 clearly show that the use of buffer memory decreases as the upper bound of traffic increases. Figure 11 indicates that minimizing traffic leads to tremendous use of the receiver's buffer memory, which we consider to be unrealistic. Thus, our modification to allow utilization of the available bandwidth (Figs. 12 and 13) will be extremely useful. If we dynamically assign the available bandwidth for each content delivery according to each request rate, we will further reduce the use of the receiver's buffer memory.

Fourth, our simulation results show that the traced traffic does not remain strictly within the available bandwidth, because our traffic-control algorithm is based on "statistical control". As a future subject for study, we must consider setting the upper bound of traffic somewhat lower than the actual available bandwidth.

6. Conclusions

Effective distribution for Personalized Media Steam Delivery is feasible by using asynchronous multicasting and layered multicasting with the improvements described in this paper. Asynchronous multicasting with our statistical traffic-control algorithm enables multicasting of temporally personalized content, which allows us to control the load on the network at will and reduce the use of the receiver's buffer memory by making full use of the available network resources. Layered multicasting with our extension to the diverse media segment and object delivery enables multicasting for spatially personalized content distribution, and we have shown the way of reducing the number of multicast groups mapped onto each media segment through co-functioning asynchronous multicasting.

To test the implementability of these techniques, we designed, as a representative example, a centralized-control network using MPLS technology. Our numerical analysis showed that this designed network enabled significant retrenching of traffic as compared with that in unicasting. Our simulation results confirmed the validity of the numerical calculation in this paper and showed the effectiveness of our statistical traffic control.

In future studies, we must study a simpler network model that allows implementation of our multicast scheme. We will explore the possibility of an autonomous distributed-control network. Our final goal is to realize Personalized Media Steam Delivery through real multicast network systems.

References

1) Miles, P.: Internet World Guide to Webcasting, John Wiley & Son (1998).
2) Nakagawa, S. and Katsumoto, M.: The future of digital media by IP communication, *IPSJ Magazine*, Vol.41, No.12 (Dec. 2000).
3) Sato, K. and Katsumoto, M.: A proposal for personalized media stream delivery, *ISPJ Report*, DPS103-14 (Jun. 2001).
4) Woo, H. and Kim, C.K.: Multicast scheduling for VOD services, *Multimedia Tools and Applications*, Vol.2, No.2, pp.157–171 (Mar. 1996).
5) Kalva, H. and Fuhrt, B.: Techniques for improving the capacity of video-on-demand systems, *Proc. 29th Annual Hawaii International Conference on System Sciences*, pp.308–315, Wailea, HI, USA, IEEE Computer Society Press (Jan. 1996).
6) Cater, S.W. and Long, D.E.: Improving video-on-demand server efficiency through streaming tapping, *Proc. International Conference on Computer Communication and Networks*, Las Vegas, NV, USA, pp.200–207 (Sep. 1997).
7) Uno, S., Tode, H. and Murakami, K.: Performance evaluations on video distribution system with multicast and burst transmission, *IEICE Technical Report*, IN99-82 (Nov. 1999).
8) McCanne, S., Jacobson, V. and Vetterli, M.: Receiver-driven layered multicast, *Proc. ACM Sigcomm* (1996).
9) Suzuki, T., Mimura, T., et al.: Development of middleware for multi QOS over IP network, *Proc. 2000*, TAO (Jun. 2000).
10) Blake, S., Black, D., et al.: An Architecture for Differentiated Services, RFC 2475 (1998).
11) Rosen, E., Viswanathan, A. and Callon, R.: Multiprotocol Label Switching Architecture, RFC3031 (2001).
12) Le Faucheur, F., Wu, L., et al.: MPLS Support of Differentiated Services, IETF Internet Draft (Feb. 2001).
13) Jacobson, V., Nichols, K., et al.: An Expedited Forwarding PHB, RFC2598 (1999).
14) Ooms, D., Sales, B., et al.: Framework for IP Multicast in MPLS, IETF Internet Draft (Jan. 2001).
15) Schulzrinne, H., Cater, S., et al.: A Transport Protocol for Real-Time Applications, RFC1889 (1996).
16) Delivery Multimedia Integration Framework, ISO/IEC FDIS 14496-6.
17) Schulzrinne, H., Rao, A., et al.: Real Time Streaming Protocol, RFC2326 (1998).
18) Singer, D., Lim, Y., et al.: A Framework for the Delivery of MPEG-4 over IP-Based Protocols, IETF Internet Draft (Nov. 2000).

Appendix

A.1 Protocol Design

Based on our proposed network model (Sec-

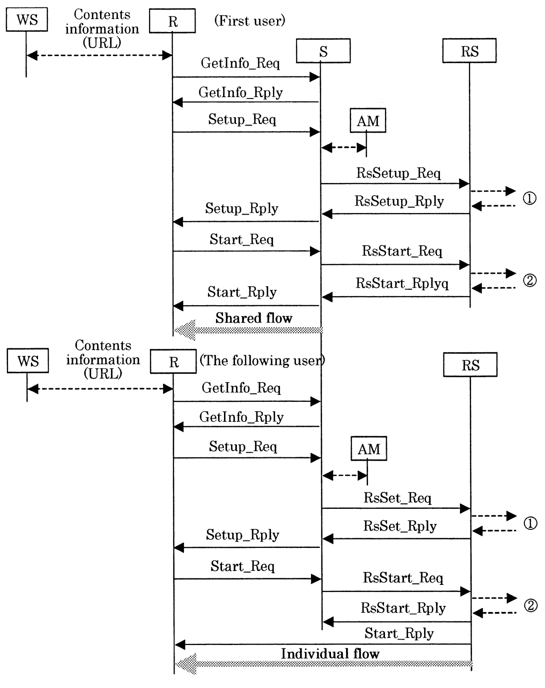

Fig. 14 Sequence between end systems.

Table 1 Messages and parameters.

Messages (meaning)	Parameters
GetInfo_Req (request program information)	URL
GetInfo_Rply (reply to program information request)	Program descriptor, media descriptor, IOD (Initial Object Descriptor)
Setup_Req (request to set up transport)	Transport protocol, unicast address, port number, user descriptor, system descriptor
Setup_Rply (reply to set up transport request)	Scenario descriptor (multicast address, port number, user-group, QOS class), session identifier
RsSetup_Req (request to setup RS)	Scenario descriptor (multicast address, port number, user-group, QOS class), flow type, session identifier
RsSetup_Rply (reply to setup RS request)	Session identifier
Start_Req (request to start)	Session identifier, period
Start_Rply (reply to start request)	Session identifier, The number of segments between the two flows
RsStart_Req (request to start RS)	Session identifier, period
RsStart_Rply (reply to start RS request)	Session identifier
SetCall_Req (request to establish call)	Session identifier, time-scaled traffic descriptor
SetCall_Rply (reply to establish call request)	Session identifier, call identifier
Data_Req (request to start sending data)	Call identifier
Data_Rply (reply to start sending data request)	Session identifier
Admis_Req (request admission control)	Call identifier, time-scaled traffic descriptor
Admis_Rply (reply to admission control request)	Call identifier, reservation identifier
Reserve_Req (request to reserve resources)	Reservation identifier
Reserve_Rply (reply to reserve resources request)	Call identifier, path-node list descriptor
Distrib_Req (request distribution)	Call identifier, distribution sequence number, path node list descriptor
Distrib_Rply (reply to distribution request)	Call identifier, distribution sequence number
SetLabel_Req (request to set labels)	Node identifier, flow identifier, forwarding cache entry, label
SetLabel_Rply (reply to set labels request)	Node identifier, flow identifier

tion 4.3), a basic sequence between end systems is shown in **Fig. 14** and **Table 1**. First, the R is aware of the presence of content and obtains the location of the S from a WS (Web Server). The R requests that the S provide some basic information about the content (media type, etc). Next, the R requests that the S set up transport functions, and the S selects the sequence patterns of segments (i.e., the user group) according to the user's specification, maps a multicast address onto each segment, and determines whether the content will be sent by a shared flow only or by both a shared and an individual flow. The S then sends these sets of information to the RS in a request to set up a relay point. The RS requests that the CC perform admission control and the RS replies to the S if the task has been done. The S returns the response for the request to set up transport functions to the R. Next, the R requests that the S start transmitting data, the S issues the same request to the RS, and the RS requests that the CC reserve resources and set paths. Then the CC replies to the RS and the RS replies to the S. If the content is sent by a shared flow only, the S returns the response for the request to start transmitting data to the R and transmits the streaming data by multicasting. If the content is sent by both a shared flow and an individual flow, the RS returns the response for the request to start transmitting data to the R and transmits the streaming data by unicasting.

A basic sequence among network node systems is shown in **Fig. 15** and Table 1. Having received a request to set up a relay point, the RS requests that the CC establish a call, and the CC requests that the TM compute the path route and perform admission control. Having received a request to start transmitting data, the RS notifies the CC, and the CC requests

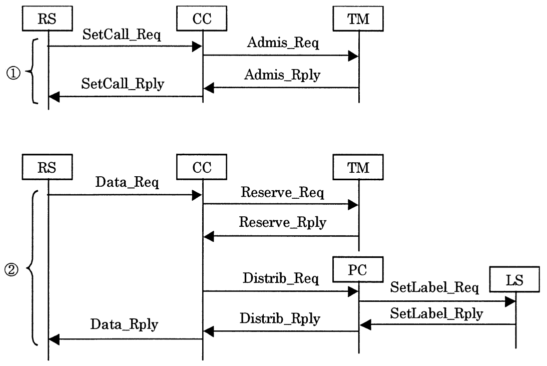

Fig. 15 Sequence among network node systems.

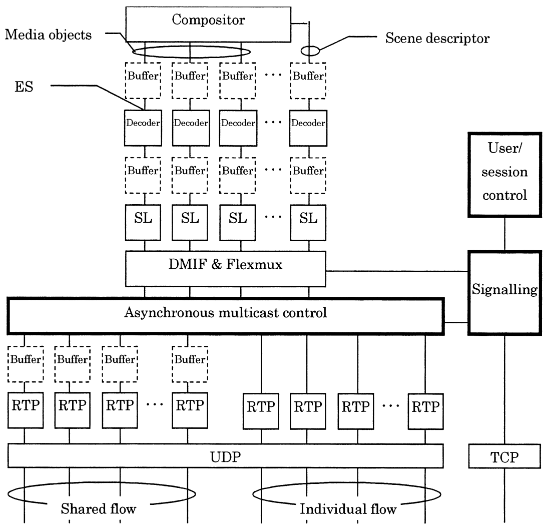

Fig. 16 Receiver system model.

that the TM reserve resources and provides the PC with path information computed by the TM. The PC requests that LSs along the path establish an LSP.

A.2 Receiver System Model Design

The receiver system model (**Fig. 16**) is based on a standard decoder model specified in MPEG-4. Each ES (Elementary Stream) for media objects and the scene descriptor corresponds to one RTP (Real-time Transport Protocol) session[15]. Functions higher than DMIF (Delivery Multimedia Integration Framework)[16] are the same as in the standard model.

The received ESs in an individual flow are decoded and played back as soon as they arrive. The ESs in a shared flow are buffered until all the ESs in an individual flow are completely played back. A coordinate function called "asynchronous multicast control" is installed between the RTP and the DMIF. The time stamp used for MPEG-4 is calculated from the time stamp in an RTP packet and the number of segments between the two flows, which is a parameter included in the message 'Start_Rply' from the RS to the R.

(Received April 3, 2002)
(Accepted November 5, 2002)

Editor's Recommendation

The authors propose two multicast techniques for personalized media stream delivery. One is an asynchronous multicast technique where an algorithm for flexible bandwidth control has been proposed. The other is a layered multicast technique for delivering diverse media segments/objects. The authors mix those techniques to achieve efficient network resource utilization. Some experimental results are also reported to show the applicability for practical porposes.

(Chairman of SIGDPS, Teruo Higashino)

Katsuhiko Sato received the B.E. degree in the Department of Physics from Aoyama Gakuin University in 1992. He have been working for Japan Radio Co., Ltd and developed Frame Relay Router, ATM Switch, W-CDMA BTS, Multi-layer Switch, etc. Since 2002 he has been a visiting researcher at Communication Research Laboratory and working toward the Ph.D. in The University of Electro-Communication.

Michiaki Katsumoto received the B.S., M.S. and Ph.D. degrees from Toyo University in 1991, 1993 and 1996 respectively. He is working now Communications Research Laboratory. His research interests include next generation Internet applications and high-quality multimedia contents. He is a member of IPSJ, IEICE, IEEE Computer Society and ACM.