A Selection Technique for Replicated Multicast Video Servers

Akihito Hiromori
Graduate School of Eng. Sci., Osaka University
hiromori@ics.es.osaka-u.ac.jp

Hirozumi Yamaguchi
Graduate School of Info. Sci. and Tech., Osaka University
h-yamagu@ist.osaka-u.ac.jp

Keiichi Yasumoto
Graduate School of Info. Sci., Nara Inst. of Science and Technology
yasumoto@is.aist-nara.ac.jp

Teruo Higashino
Graduate School of Info. Sci. and Tech., Osaka University
higashino@ist.osaka-u.ac.jp

Kenichi Taniguchi
Graduate School of Info. Sci. and Tech., Osaka University
taniguchi@ist.osaka-u.ac.jp

Abstract

In this paper, we propose a selection technique for replicated multicast video servers. We assume that each replicated video server transmits the same video source as different quality levels' multicast streams. Using an IGMP facility like mtrace, each receiver monitors packet count information of those streams on routers and periodically selects the one which is expected to provide low loss rate and to be suitable for the current available bandwidth of receivers. Moreover, collection of packet count information is done in a scalable and efficient manner by sharing the collected information across receivers. Our experimental results using the network simulator have shown that our method could achieve much higher quality satisfaction of receivers, under the reasonable amount of tracing traffic.

1 Introduction

According to the recent progress of high-speed networks, we can expect that a large number of multimedia contents, especially video contents will be distributed through networks in near future. In general, such multimedia contents consume large amount of bandwidth, therefore the network may be congested if a lot of clients access to a single server. Server replication is one effective solution for such a problem. Service providers can dissolve the network congestion and the convergence of access to a single server, and clients can select suitable servers depending on the network status.

There are a lot of studies investigating how clients collect the network information (bandwidth, network topology, transmission delay and so on) and how they select suitable servers [1, 2]. However, these studies assume unicast servers because their target contents are on-demand contents such as WWW documents. On the other hand, for video broadcasting on scheduled time such as Internet TV, multicast communication is useful to save bandwidth, and new research results of the selection techniques of replicated multicast servers have been proposed [3, 4, 5]. However, these studies assume that much information should be known by each receiver such as (a part of) network topology and available bandwidth on each link, and do not mention how to collect the information. Thus it is not straightforward to adopt them on IP multicast networks.

In this paper, we assume that replicated multicast servers are located on different nodes, and each server transmits the same video source into different quality levels' multicast streams (*i.e.* independent multicast streams of different rates). Under this assumption, we propose a new server selection technique where each receiver monitors packet count information (the numbers of forwarded packets) of those streams on routers using IGMP mtrace query [6] and periodically selects the one which is expected to provide low loss rate and to be suitable for the available bandwidth of the receiver. Furthermore, our selection technique enables receivers to share packet count information in an efficient and distributed manner, in order to avoid significant impact to network load caused by the flood of mtrace query messages sent from a large number of receivers periodically.

We have evaluated the performance of our method using the network simulator *ns-2*[7]. The experimental results have shown that our method could achieve much higher satisfaction in terms of average *quality values* at receivers, keeping the amount of control traffic low enough.

2 Design Concept

For replicated multicast video servers on IP networks, we think that the followings should be considered, (a) how to collect network status, (b) how to deal with heterogeneous receivers and bandwidth fluctuation, and (c) how to keep scalability for a large number of receivers. Regarding (a), some selection methods for unicast servers measure RTT to estimate end-to-end delay and jitter. However, in multicast communication, a delivery tree has been already constructed when a receiver tries to join a group, and the receiver may be able to select the best one by obtaining information about the existing multicast trees in advance. Our method uses IGMP mtrace facility for this purpose. Note that some other methods use this facility for other purposes, for example, Ref. [8] uses it to find paths for local error recovery in reliable multicast. Regarding (c), monitoring routers by a large number of receivers may yield a scalability problem. Considering this trade-off between (a) and (c), we address how to reduce monitoring costs in Section 4. (b) is a common and well-known issue in multicast communication where some receivers in a group using low-speed links may not receive the content while others in the same group may not efficiently use their high-speed links. In this paper, *simulcasting*[9] is assumed where a server is capable of providing a single video source as different quality's streams, and allows receivers to switch quality levels as well as switching servers for rate adaptation. Note that a more sophisticated scheme for the heterogeneity has been known as Receiver-driven Layered Multicast (RLM)[10]. The possibility of RLM in the replicated server architecture is discussed in Section 6.

3 Stream Selection Algorithm

3.1 Replicated Server Architecture

We consider networks where multiple servers on different locations distribute the same video content. Each server encodes the video content into independent L streams, whose *quality levels* are different from each other (e.g., w.r.t. spatial and temporal resolution). We denote each quality level as an integer number where 1 corresponds to the lowest quality and L does the highest quality. Each server transmits the data streams of the video content with $1...L$ quality levels via independent L multicast groups. Hereafter, we call a data stream transmitted from server S_i with quality level l simply as a *stream*, and denote it as $st_{i,l}$.

3.2 Monitoring Multicast Packet Counts

Most of current IGMP supported routers implement tracing facility of IGMP as specified in [6]. Several tools for monitoring IP multicast traffic using this facility have been proposed so far (see [11] for survey) and among them there is a tool called *mtrace*[6]. Using mtrace with a multicast group address and a source host address, we can obtain a sequential list of routers' addresses on the path from the sender to the receiver with the total numbers of forwarded packets and time stamps on those routers (called *packet count information*).

Multicast routers count the total numbers of forwarded packets during their operation time for every stream(group). We assume that receivers periodically send mtrace query messages for each group. Using the last two query results which include routers' packet counts and their time stamps, the number of forwarded packets per second at each router can be computed. Consequently, each receiver can know the delivering path from a server to the receiver and the number of forwarded packets per second (packet count information) on the intermediate routers of the path.

3.3 Receivers' Knowledge

We assume that each receiver R_j knows the followings as either given information in advance or obtained information by monitoring.

- $path_{i,j}$; the delivery path from server S_i to R_j. We assume that R_j knows $path_{i,j}$ for each server S_i. This is obtained by one time execution of mtrace query for each server.
- $gr_{(i,l),j}$; the *grafting router* (branch router) of $st_{i,l}$ for R_j. It is the router on $path_{i,j}$ that receives $st_{i,l}$ and is the nearest to R_j. We assume that R_j knows $gr_{(i,l),j}$ for each $st_{i,l}$. This router is found as the last router on $path_{i,j}$ where the packet count of $st_{i,l}$ is not zero.
- $ratio_{(i,l)}@r$; the packet arrival ratio of $st_{i,l}$ at router r. We assume that R_j knows $ratio_{(i,l)}@r$ for each pair of $st_{i,l}$ and router r on $path_{i,j}$. Hereafter, $count_{(i,l)}@r$ denotes the number of packets per second at router r obtained by the periodic execution of mtrace queries. $ratio_{i,l}@r$ can be computed by $\frac{count_{(i,l)}@r}{count_{(i,l)}@S_i}$. Note that $count_{(i,l)}@S_i$ denotes the number of transmitted packets of stream $st_{i,l}$ at server S_i and we can not measure this packet count by mtrace. However, this is equal to the number of forwarded packets at the router on the same network as S_i if we assume that packets are rarely discarded on this router. In general,

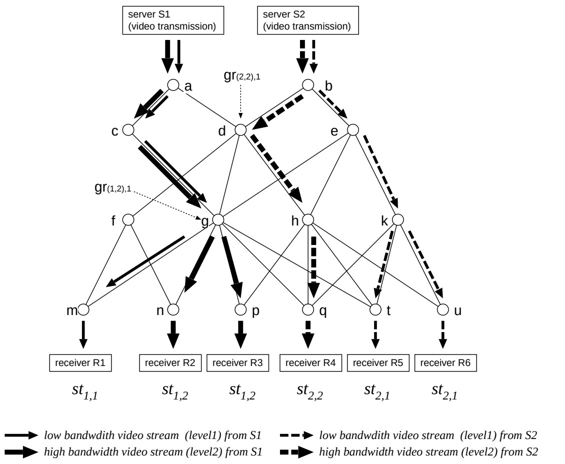


Figure 1. Selecting Streams: Example

congestion rarely occurs at such a router, since there are only a few senders in usual. Therefore this assumption is reasonable.

- $ratio_{(i,l)}@R_j$; the packet arrival ratio of $st_{i,l}$ at receiver R_j where R_j currently receives $st_{i,l}$. $ratio_{(i,l)}@R_j$ can be computed by $\frac{count_{(i,l)}@R_j}{count_{(i,l)}@S_i}$.

3.4 Selecting Streams : Overview by Example

In our technique, each receiver dynamically selects a stream (determines a pair of a server and a quality level) using the packet arrival ratio at each multicast router on the delivery paths from servers to the receiver.

In Fig. 1, two servers S_1 and S_2 transmit the same video content via multicast streams of two different quality levels: level1 (low) and level2 (high) (thus $i = 1, 2$ and $L = 2$). Each receiver $R_j(j = 1..6)$ is receiving one of those multicast streams. For example, R_1 is receiving stream $st_{1,1}$ (the stream of level1 from server S_1) and R_5 is receiving stream $st_{2,1}$ (the stream of level1 from server S_2).

Here, given the packet count information, we think that the following two factors can be used to infer a "good" stream for each receiver; packet arrival ratio at a grafting router (branch router) and the number of hops between the grafting router and the receiver. Now let us suppose that receiver R_1 has been receiving stream $st_{1,1}$ stably for a while (*i.e.*, the packet arrival ratio $ratio_{(1,1)}@R_1$ is high). In order to receive the stream of the higher level (level2), R_1 finds the grafting routers, $gr_{(1,2),1}$ (router g) and $gr_{(2,2),1}$ (router d) ($path_{1,1}$ = *a-c-g-m* and $path_{2,1}$ = *b-d-f-m*). Also let us assume that both $ratio_{(1,2)}@gr_{(1,2),1}$ and $ratio_{(2,2)}@gr_{(2,2),1}$ (packet arrival ratios at grafting routers g and d, respectively) are higher than a certain threshold. The threshold is a lower bound of packet arrival ratio which is expected to provide stable quality. R_1 selects one of the two streams $st_{1,2}$ and $st_{2,2}$, with minimal number of hops to the grafting router. In this example, since the distance from the grafting router of stream $st_{1,2}$ to R_1 is 1 hop and that of $st_{2,2}$ is 2 hops, R_1 selects $st_{1,2}$.

Similarly, if the packet arrival ratio at a receiver becomes lower than a certain threshold, the receiver selects one of streams of the same or the lower quality levels which are expected to be stable. For example, if $ratio_{(1,2)}@R_3$ becomes lower than a certain threshold, R_3 tries to receive one of $st_{2,2}$, $st_{1,1}$ and $st_{2,1}$. If their packet arrival ratios at their grafting routers are relatively good, R_3 selects $st_{2,2}$ in order to keep the current quality level.

3.5 Selection Algorithm

In our selection procedure, each receiver R_j periodically tries to select a new stream (or keeps receiving the current stream), according to the packet arrival ratio of its receiving stream $st_{i,l}$ ($ratio_{(i,l)}@R_j$).

In selecting streams, each receiver first decides a new quality level suitable than the current level. RLM[10] uses *join-experiment* to decide the number of layers (*i.e.*, receiving rate) to subscribe. *join-experiment* lets a receiver attempt to subscribe the higher layer if she/he wants, and unsubscribe it if significant packet loss is experienced. Additionally, the receiver uses a *join-timer* for every layer, which suggests a time period for a next join-experiment of the layer. This period is increased if a join-experiment is failed, in order to prevent frequent experiments that are likely to fail. Even though our method is different from RLM (we assume independent streams rather than layers), we believe that this scheme is useful for congestion control in our case. Therefore a similar policy can be applied to determine a new quality level in selecting a stream.

In order to describe the selection behavior of a receiver, we define the following three states, (1) *level-up* state, (2) *stable* state and (3) *level-keep-or-down* state, according to $ratio_{(i,l)}@R_j$.

1. A receiver R_j where $ratio_{(i,l)}@R_j \geq P_1$ ($0 < P_1 \leq 1$) is regarded to be in *level-up* state. P_1 is a lower bound ratio by which R_j can infer that there is not significant loss near R_j. In simulation, we have used $P_1 = 0.95$. This value, which is close to 1.00, may yield the better results in our experience, because $ratio_{(i,l)}@R_j$ is likely to fall down greatly (down to about 0.80 or lower) in the event of congestion, and to be close to 1.00 otherwise. R_j selects a new stream $st_{i',l+1}$ if (a) the join-timer expires, (b) $ratio_{(i',l+1)}@gr_{(i',l+1),j} \geq P_2$ holds (this

means that the packet arrival ratio at the grafting router is not less than P_2, and is explained later) and (c) the hop count from $gr_{(i',l+1),j}$ to R_j is the minimum of all the other streams of quality level $l+1$. If there is no such a stream $st_{i',l+1}$, R_j keeps receiving $st_{i,l}$. If this experiment fails (*i.e.* the receiving rate is instable and R_j is back to the current level), R_j extends the join-timer.

Even though the quality level of the new stream is one level higher than the previous stream, there is the possibility that R_j cannot gain the expected quality due to the traffic on the links from S_i to $gr_{(i',l+1),j}$, and also the traffic on the links from $gr_{(i',l+1),j}$ to R_j. By selecting a stream where a certain packet arrival ratio has been already achieved at the grafting router and the distance between the grafting router and the receiver is the shortest of all such streams, the new stream with one level higher is likely to be received stably.

2. A receiver R_j where $P_1 > ratio_{(i,l)}@R_j \geq P_2$ ($P_1 > P_2 > 0$) is regarded to be in *stable* state. P_2 is a lower bound ratio under which R_j cannot tolerate the damage caused by packet loss and by which R_j considers that congestion occurs. In the simulation, we have used $P_2 = 0.85$.

   In this case, R_j keeps receiving $st_{i,l}$.

3. Each receiver R_j where $P_2 > ratio_{(i,l)}@R_j$ is regarded to be in *level-keep-or-down* state.

   In this case, R_j selects a new stream $st_{i',l'}$ ($l' \leq l$) where $ratio_{(i',l')}@gr_{(i',l'),j} \geq P_2$ (the packet arrival ratio at the grafting router is not less than P_2) and l' is the highest level of all such streams. If such a stream is not uniquely determined, R_j selects the one where the hop count from the grafting router $gr_{(i',l'),j}$ to R_j is minimum.

   For streams whose packet arrival ratios at the grafting routers are not less than the threshold P_2, R_j selects the one whose level l' ($l' \leq l$) is the highest of all such streams. In case of $l' = l$, R_j can keep the current quality level, otherwise the quality level is lower than the current level, however, the actual quality is expected to be improved.

4 Improving Scalability in Monitoring Multicast

As stated in the previous section, our technique is a monitoring based approach. Here a reasonable question is that, we may experience the implosion of mtrace packets near a sender, like the NAK implosion problem in feedback-based reliable multicast.Here is a simple analysis of the amount of mtrace packets. Assume that there are 1000 receivers and they periodically send mtrace query packets (at most 0.2KB) for every 2 seconds, for a pair of a multicast group and its sender. These packets are forwarded toward the sender, and thus aggregated at the sender. If each server distributes three streams($L = 3$), the amount of mtrace traffic at each server is $\frac{0.2*8*1,000*3}{2} = 2.4Mbps$. Moreover, the router on the root of the tree needs to process 500 queries per second. Needless to say, these values are impractical and intolerable, and thus some efficient way is desired.

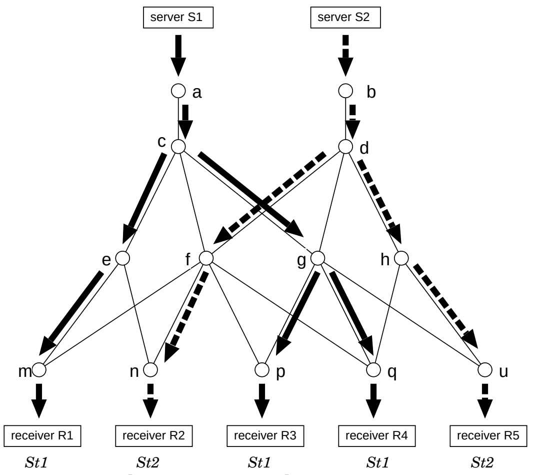


Figure 2. Multicast Trees.

In order to avoid such message explosion, we roughly control receivers in a distributed manner in order to limit the number of queries on networks, and enable to share packet count information. The idea is as follow. We use a multicast group (say C) as a control group where all the receivers are the members. Also every receiver has his/her own timer. One of the servers sends a signalling message to C for every T_m period, for loosely synchronization among receivers. Each receiver sets a random value to the timer and then starts it whenever it receives a synchronization message on C. The random value is determined based on the exponential distribution with parameter λ. When the timer expires, it sends a query message only to the current receiving stream and sends the received response (query result) to C, unless it has received query results from others on C. This enables receivers to share packet count information of the routers on the shared part of delivering paths (*e.g.* if two receivers A and B share a part of their paths to a server and B executes a query, the query result includes packet count information of the routers on the shared part of the paths) and thus the number of queries can be reduced. Due to the limitation of space, we omitted formal description of the method. Readers may refer [12] for the details.

We explain this idea using Fig. 2. There are two servers S_1 and S_2, and they send streams st_1 and st_2, respectively. Receivers R_1, R_3 and R_4 receive st_1, therefore they can send mtrace queries only to st_1 (R_2 and R_5 can do only to st_2). For every T_m, some of R_1, R_3 and R_4 can send mtrace queries to S_1, and now assume that only R_3 and R_4 send queries in this period. Then they know the latest packet counts of the routers on their paths (a-c-g-p and a-c-g-q), and send these query results to the control group C. On the other hand, R_1 can know the latest information on routers a and c which are on its path (a-c-e-m), however cannot know those on e and m. In this case R_1 uses the last information which R_1 knows. Using a random delay generator based on an exponential distribution, the information on the path of each receiver will be updated eventually. As clearly known, the information of a router closer to a sender is updated more frequently at each receiver.

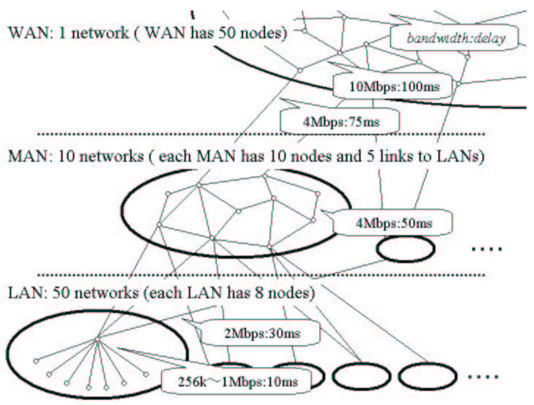
For the information of streams which R_j does not currently receive, R_j knows their information by listening to the control group C. Since R_j does not send mtrace queries to those streams, it never knows the information on the routers which are on the paths to their senders but not included in the multicast trees. For example, R_2 and R_5 send mtrace query results to C and those results do not contain information on m. However, R_1 can know the information of the routers b, d and f which are on the path $path_{2,1}$, and R_1 knows that all the information which is needed to execute our selection procedure (*e.g.*, the grafting router is found as the router which is the closest to the receiver on the path from the server to the receiver and is included in at least one of mtrace query results sent to C).

5 Experimental Results

In order to evaluate the performance of our proposed method, we have designed and implemented a protocol for collecting the number of arrival packets at intermediate routers, as a module of network simulator *ns-2*[7]. This protocol works in the network layer and is based on the scheme of IP multicast tool mtrace.

In Section 5.1, we have evaluated receivers' *quality satisfaction* which we think is the most important factor in video distribution. Then in Section 5.2 the influence of topologies on the quality satisfaction has been examined. Finally, in Section 5.3, the amount of control traffic (mtrace queries and results) has been measured.

A number of researches have been investigated for modeling traffic patterns in the Internet [13], however, it is still a challenging task. In this paper, we have roughly distinguished the characteristics of background traffic into two typical categories (a) stable case and (b) bursty case, and we have carried out simulations under both cases. In case (a), we have generated no connection. In case (b), we have generated 30 unicast connections of constant bit rates (about 2 Mbps). Their alive time and locations were determined randomly.


WAN: 1 network (WAN has 50 nodes)
bandwidth:delay
10Mbps:100ms
4Mbps:75ms
MAN: 10 networks (each MAN has 10 nodes and 5 links to LANs)
4Mbps:50ms
LAN: 50 networks (each LAN has 8 nodes)
2Mbps:30ms
256k~1Mbps:10ms
Figure 3. Network Topology

As comparison, we consider a method where the nearest server is allocated in advance for each receiver (thus he/she can only select quality levels). This method is called a *fixed server method*. Quality level selection policy in the fixed server method is the same as our method.

We have used networks of a hierarchical topology model called Tiers [14]. The tiers model consists of three types of organizations, WAN, MAN and LAN. In our simulation, we have determined the numbers of nodes in WAN, MAN and LAN as 50, 10 and 8, respectively. Also, the total numbers of WAN, MAN and LAN are 1, 10 and 50, respectively (thus the number of nodes in a network is 550). LAN has a star topology. We have determined link capacities and link delays as specified in Fig. 3.

Servers are located on WAN and receivers are located on LANs. Each server sends three streams ($L = 3$, three quality levels) whose transmission rates are 256kbps, 512kbps and 1Mbps. Also, we have used DVMRP as a multicast routing protocol.

5.1 Receivers' Quality Satisfaction

We define a *quality value* to measure the quality satisfaction of receivers. It represents the quality of video that a receiver receives. If a receiver has received a stream $st_{i,l}$ during a time period (let t_a and t_b denote its starting time and ending time, respectively) and if its packet arrival ratio at time t is p_t, we define the quality value during the time period as the integral of the product of quality level l and arrival ratio p_t (*i.e.*, $\int_{t_a}^{t_b} l * p_t dt$). We have measured average quality values of all the receivers during the simulation time. Note that a quality value may become high not only in the case that the packet arrival ratio is stable and its quality level is suitable but also in the case that the quality level is too high and a lot of packet losses are experienced. In general, quality satisfaction of receivers in the latter case is considered low. Therefore, we also show packet arrival ratios in this section.

Average Quality Value We have measured average quality values in (a) stable cases and (b) bursty cases. Fig. 4 shows those values in our method and the fixed server method. The numbers of receivers are 50, 100 and 150. The vertical axis represents average quality values and the horizontal axis represents simulation cases. Note that in Fig. 4 (a), the theoretically maximum quality values are calculated shown[1].

In stable cases (in Fig 4 (a)), average quality values in our method are around 95 % of those in the fixed server method. This is because of the existence of control traffic (mtrace queries). However, comparing quality values in our method with maximum quality values, our method could achieve 85 % - 95 % (93% in average) of the maximum quality values.

On the other hand, in bursty cases (in Fig 4 (b)), average quality values in our method exceed those in the fixed server method in almost all cases (120% - 140% compared with the fixed server method). Considering the results in both cases, we can see that our method could nicely avoid congested paths.

Average Packet Arrival Ratio In Fig. 5, the average packet arrival ratios in the same experiments above are shown.

Average packet arrival ratios in both methods are similar, and greater than 90 % in stable cases (Fig. 5 (a)). In bursty cases (Fig. 5 (b)), although the ratios were greater than 90 % as in the stable cases, the fixed method could not keep 90%.

From the above results, receivers could select higher quality levels' streams in our method than the fixed server method and avoid congested paths in the event of congestion.

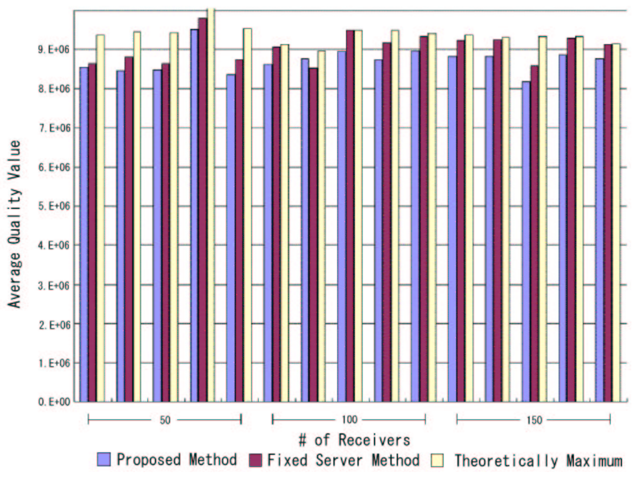
5.2 Network Topology Influence

In our method, receivers select streams based on the two factors: packet arrival ratios at branch routers (grafting routers) and the numbers of hops from them. Even though our method is not designed for specific topology models, the latter factor may differ in different topology models. Therefore, in order to confirm that our selection algorithm works well under typical network topologies, we have measure average quality values on mesh-like topology networks and pure-tree topology networks as well as on Tiers model (hierarchical topology networks). Note that on pure-tree networks, servers are located near the root node, while they are randomly distributed on mesh-like networks. The results are shown in Fig. 6.

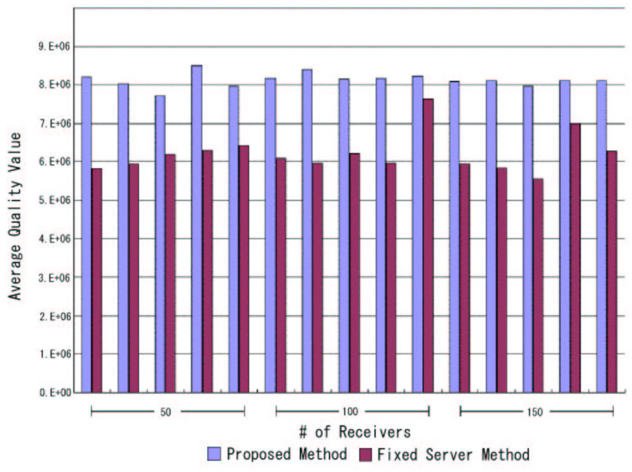
On pure-tree networks, delivering paths from servers are shared in most cases. For this reason, our method and the fixed server method fell into the similar results. On the other hand, on Tiers networks which we have used in the experiments in the previous section, since receivers could have a few different paths, our selection has achieved better performance than the fixed server method by selecting other servers in the event of congestion.

We can find much more striking difference between our method and the fixed server method on mesh-like

[1]We have calculated the (theoretically) maximum of average quality values using an integer linear programming (ILP) technique under a fixed allocation of streams to receivers.

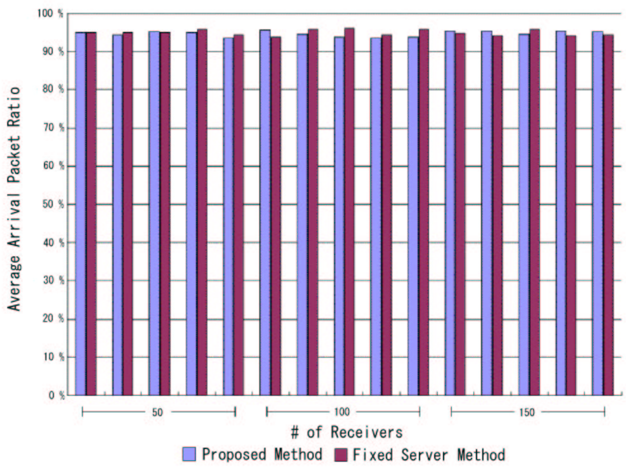


(a) under stable background traffic

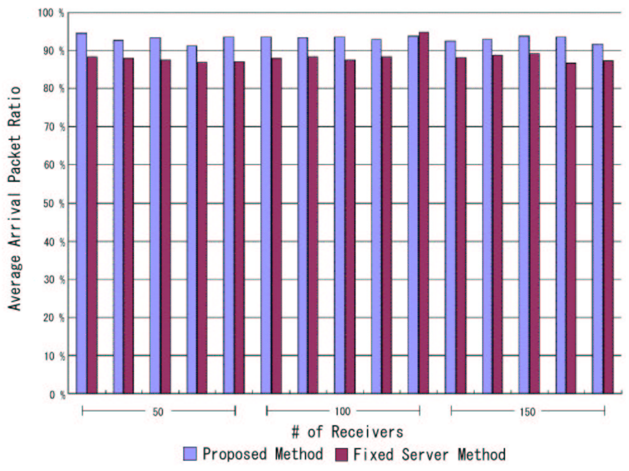


(b) under bursty background traffic

Figure 4. Average Quality Value



(a) under stable background traffic



(b) under bursty background traffic

Figure 5. Average Packet Arrival Ratio

topology networks. As we know, since receivers could have a variety of paths, they could adaptively select a feasible server in our method.

5.3 Control Traffic

In Sections 5.1 and 5.2, we have confirmed that our method could totally archive better performance than the fixed server method even in large-scale networks (550 nodes). However, as we stated in Section 4, the number of mtrace queries increases depending on the number of receivers. Even though we have proposed an idea to loosely control the number of queries transmitted, we should confirm that it can actually keep the amount of mtrace query traffic low enough.

In the experiments in Sections 5.1 and 5.2, a server sent a synchronization message for every 1 second, and receivers were controlled where only 25 % of them sent mtrace query in 1 second ($\lambda = 0.3$ in the exponential distribution used to generate random timer values at receivers). Using this value of $\lambda = 0.3$, we have also carried out experiments to measure the amount of control traffic (mtrace query traffic and traffic on the control group C) varying the number of receivers.

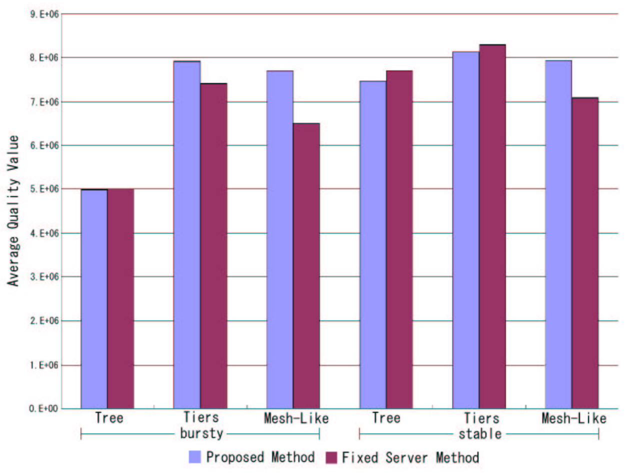


Figure 6. Average Quality Value on Different Networks

The result is shown in Fig. 7. In this figure, the measured control traffic at a server and a receiver is depicted. The vertical axis represents the number of receivers and the horizontal axis represents the amount of the control traffic. The result has shown that the amount of traffic in both server and receiver sides is less than 12 kbps even in 250 receivers.

Note that we have also proposed a technique to autonomously adjust the control traffic under a certain amount, independent of the number of receivers (see Ref. [12] for details of dynamic λ control). Fig. 8 shows how the technique has automatically adjusted the amount of control traffic in a simulation case. We have examined two cases where λ was fixed to 0.75 in one case, and in another case λ was controlled (initially set to 0.75) so that the traffic could be less than 4kbps. In Fig. 8 we can see that the amount of control traffic is nicely controlled by our dynamic λ control technique.

6 Conclusion

In this paper, we have proposed a new selection technique for replicated video multicast servers. Under the assumption that there exist replicated video servers and each server transmits the same video source as different multicast streams, our technique allows each receiver to monitor packet count information on routers using an IGMP facility and to periodically select the one which is expected to provide low loss rate and to be suitable for the current available bandwidth of the

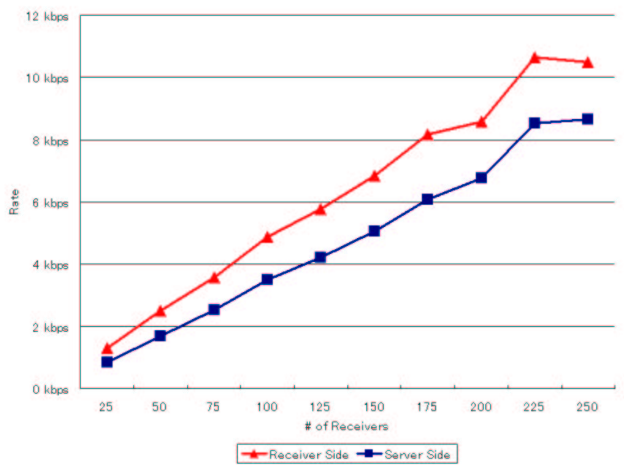


Figure 7. Control Traffic($\lambda = 0.3$)

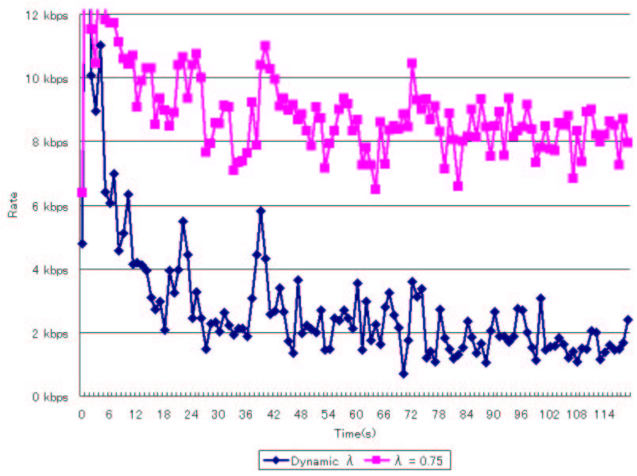


Figure 8. Time vs Control Traffic ($\lambda = 0.75$ and dynamic λ)

receiver.

As briefly mentioned in Section 1, Receiver-driven Layered Multicast [10] on replicated server architecture is also considerable, and is a challenging issue. A receiver is possible to select different servers for subscribing his/her layers, whereas it should consider the differentiation of path characteristics as between layers. This is part of our future work.

References

[1] M. Sayal, Y. Breitbart, P. Scheuermann, and R. Vingralek, "Selection algorithms for replicated web servers," in *Proc. of Workshop on Internet Server Performance (WISP98)*, 1998.

[2] R. L. Carter and M. E. Crovella, "On the network impact of dynamic server selection," *Computer Networks*, vol. 31, no. 23-24, pp. 2529–2558, 1999.

[3] Z. Fei, M. H. Ammar, and E. W. Zegura, "Optimal allocation of clients to replicated multicast servers," in *Proc. of 1999 Int. Conf. on Network Protocols (ICNP'99)*, pp. 69–76, 1999.

[4] G. Riley, M. Ammar, and L. Clay, "Receiver-based multicast scoping: A new cost-conscious join/leave paradigm," in *Proc. of 1998 Int. Conf. on Network Protocols (ICNP'98)*, pp. 254–261, 1998.

[5] A. Hiromori, H. Yamaguchi, K. Yasumoto, T. Higashino, and K. Taniguchi, "Fast and optimal multicast-server selection based on receiver' preference," in *Proc. of 7th Int. Workshop on Interactive Distributed Multimedia Systems & Telecommunication Service(IDMS2000)(LNCS 1905)*, pp. 40–52, 2000.

[6] W. Fenner and S. Casner, "A "traceroute" facility for IP multicast," in *Internet Draft*, 2000.

[7] B. MASH Research Group University of California, "The network simulator ns-2," 2000. http://www-mash.cs.berkeley.edu/ns/.

[8] B. N. Levine, S. Paul, and J. J. Garcia-Luna-Aceves, "Organizing multicast receivers deterministically by packet-loss correlation," *ACM Multimedia*, pp. 210–210, 1998.

[9] S. Fischer, A. Hafid, G. Bochmann, and H. Meer, "Cooperative QoS management for multimedia applications," in *Proc. of Int. Conf. on Multimedia Computing and Systems (ICMCS'97)*, pp. 303–310, 1997.

[10] V. Jacobson, S. McCanne, and M. Vetterli, "Receiver-driven layered multicast," in *Proc. of ACM SIGCOMM'96*, pp. 117–130, 1996.

[11] K. Sarac and K. Almeroth, "Supporting multicast deployment efforts: A survey of tools for multicast monitoring," *Journal of High Speed Networking*, vol. 9, no. 3-4, pp. 191–211, 2000.

[12] A. Hiromori, H. Yamaguchi, K. Yasumoto, T. Higashino, and K. Taniguchi, "A selection technique for replicated multicast video servers," tech. rep., http://www-tani.ist.osaka-u.ac.jp/, 2002.

[13] S. McCreary and K. Claffy, "Trends in wide area IP traffic patterns," tech. rep., CAIDA, 2000. http://www.caida.org/outreach/papers/AIX0005/.

[14] K. Calvert, M. Doar, and E. Zegura, "Modeling internet topology," *IEEE Communications Magazine*, vol. 35, no. 6, pp. 160–163, 1997.