# Dynamic Organization of Active Video Multicast in Heterogeneous Environment

Go Yoshida, Naoki Wakamiya, Masayuki Murata, Hideo Miyahara
Graduate School of Information Science and Technology, Osaka University
1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan
E-mail: {g-yosida,wakamiya,murata,miyahara}@ist.osaka-u.ac.jp

***Abstract*—In this paper, we introduce active network technologies to a video multicast system that can satisfy heterogeneous client requests in an efficient and effective way. We employ active nodes that adapt incoming video streams at the user's request by using transcoders or filters and then dynamically re-organize multicast sessions to accommodate clients. Simulation experiments demonstrate that our methods can appropriately split, merge, and move multicast groups to handle client-to-client heterogeneity.**

***Keywords; video multicast; active network; video filtering***

## I. INTRODUCTION

In recent years, networked video applications like video streaming, TV conferencing, and live broadcasting have become popular due to increased access-link capacity and greater computing power of individual PCs. IP multicast is one of the promising technologies through which a video source distributes video streams to a number of clients in an efficient way. To provide clients with video streams at a satisfactory level of quality, we should take into account the client-to-client heterogeneity [1]. The quality of a video stream that a client desires differs from client to client according to the available bandwidth, processing capability of the client's system, and the user's preferences for the perceived video quality.

To cope with this problem, we proposed a method of video multicast distribution with active network technology where highly intelligent intermediate nodes, called active nodes, adapt incoming video streams at the user's demands and dynamically re-organize multicast sessions [2]. In active networks, the behaviors of active nodes toward packets can be easily and flexibly tailored according to the demands of network administrators, applications, and users [3]. In our paper [2], we introduced active nodes into a structured network that consists of stub networks and a core network connecting them. A video stream is distributed over an overlay network that consists of a video server and active nodes. Active nodes in stub networks, called local servers, receive requests from clients in the stub network, organize multicast groups, and provide each multicast group with a video stream whose level of quality is adjusted to the group. While multicasting video streams, a local server dynamically re-organizes multicast groups by splitting a multicast group where the variation in clients' reception conditions is too large and by merging two multicast groups whose reception conditions are similar. Though re-organization, each client is accommodated in an appropriate multicast group and receives a video stream with a desirable level of quality.

However, splitting and merging do not necessarily lead to the improvement of the reception conditions that clients experience, since the root node of multicast groups is kept the same and paths from the local server to clients do not change much. In this paper, to achieve more efficient and effective video multicasting, we propose an additional method of re-organizing a multicast tree, i.e., a move operation. By delegating the management of a multicast group to another local server in the same stub network, the topology of the multicast tree changes so that it avoids the bottleneck links that disturbed video multicasting on the previous tree. For this purpose, a local server first investigates the possibility of improving the reception conditions and tries moving the multicast group to a better local server.

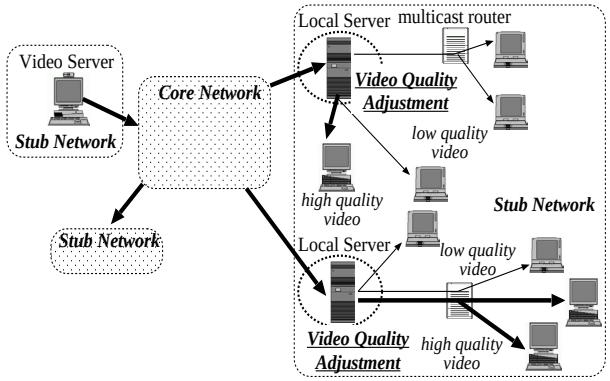


Figure 1: Active video multicast

The rest of the paper is organized as follows. In Section 2, we introduce our framework of active video multicasting. In Section 3, we first explain methods of dynamic tree re-organization on a single local server. Then the move mechanism for inter-server re-organization is described in Section 4. We show several simulation results in Section 5. Finally, we give our conclusions and describe future work in Section 6.

## II. ACTIVE VIDEO MULTICAST

Our mechanism considers a hierarchically configured network that consists of a core network and several stub networks as shown in Fig. 1. Active nodes in a stub network behave as "local servers" for clients in the domain. A local server is responsible for providing video streams of desired quality to clients, and it receives a video stream of higher quality from the originating video server or another local server through the so-called global video distribution tree. Then the local server applies video-quality adjustment to the stream in accordance with client QoS requirements and the network conditions. Local servers communicate with each other within the domain to dy-

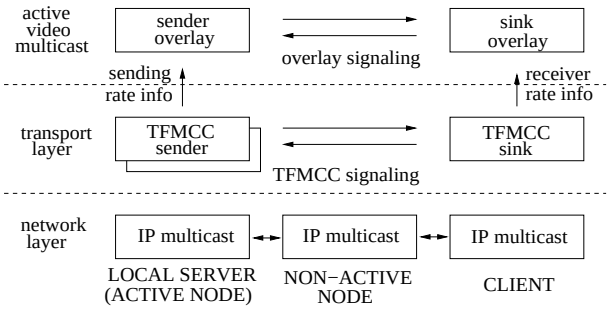


Figure 2: Layered structure of framework

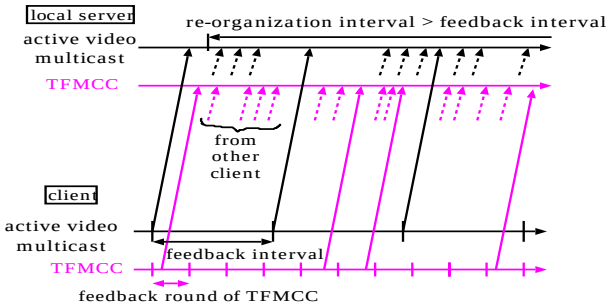


Figure 3: Control interval

namically organize appropriate local distribution trees.

First, a new video distribution service is initiated by a video server and advertised over networks. A client that intends to receive the service registers itself by sending a message to a local server in the domain. To distribute a video server in the stub network, a local server begins with only one multicast group composed of all clients that it received requests from. Then, at a regular interval, it first tries to split a multicast group into two, then attempts to merge two multicast groups into one, and finally considers moving a multicast group.

In our framework, we use TFMCC (TCP-Friendly Multicast Congestion Control) [4, 5], which is a rate control protocol to make real-time multicast applications behave in a fair manner with TCP sessions. In TFMCC, a sender initially sets the sending rate at one packet per RTT. While receiving a video stream, each receiver calculates its TCP-friendly rate from observation of the loss event rate and the RTT and reports it to the sender. The sender sets the sending rate of the multicast group at the lowest reported rate. In the case of video multicasting, a kind of video filtering method is applied to a video stream to adjust its rate to the target rate. TFMCC has a feedback suppression mechanism to prevent feedback implosion. Feedback information that reports a higher rate than the current sending rate is unnecessary for rate control. Therefore, a sender chooses a receiver that reports the worst reception condition, as CLR (Current Limiting Receiver), and allows it to send feedback preferentially at least once per feedback round. Since an instantaneous and drastic rate increase causes serious congestion, the rate of increase is limited to $\frac{8s}{R_{max}}$ [bps] every $R_{max}$ seconds, where $s$ and $R_{max}$ stand for the packet size in bytes and the maximum RTT among reported RTT, respectively.

We consider an overlay to TFMCC as shown in Fig. 2. Being a transport protocol, TFMCC relies on IP multicast for routing and group join/leave mechanisms. The TFMCC protocol runs between the local server and the receivers of each group. Each TFMCC receiver calculates its own TCP-friendly rate and then sends it to the sender to define the rate of the group. The overlay layer runs between a local server and its receivers. At the receiver side, it regularly sends the calculated TCP-friendly rates to the local server at predetermined feedback intervals as shown in Fig. 3. At the local server side, it also regularly re-organizes group membership in accordance with the received feedback at the re-organization interval. In contrast with TFMCC, which basically needs information of the worst receiver, we need rate information of all members of the group, and thus it is not possible to use feedback suppression. Nevertheless, because the control interval is longer, feedback implosion can be controlled because it is sent less frequently.

## III. Management of Group Membership

For each multicast group, a local server applies video filtering to a video stream and distributes it on a multicast tree. The quality of the video stream is adjusted according to the worst reception condition among clients so that all clients in the multicast group can receive and play it out. Therefore, when the variation in reception conditions is diverse, some clients that can potentially receive a video stream at a higher rate would not be satisfied. Taking into account such heterogeneity with the goal of satisfying clients as much as possible, a local server splits the multicast group into two so that clients with similar reception conditions are accommodated in the same multicast group. On the other hand, if there are two multicast groups whose sending rates are close to each other, a local server merges them into a single group to avoid waste in the bandwidth and reduce the server load.

In this section, we explain two methods for management of group membership: split and merge.

### A. *Splitting the Multicast Group*

A multicast group is split in two if the variation in the reported TCP-friendly rate values is too large. For this purpose, we use the variation coefficient $C_i = \frac{\sigma_i}{\gamma_i}$ of a multicast group $G_i$ $(1 \leq i \leq M)$, where $\sigma_i$ is the standard deviation and $\gamma_i$ is the average of the reported rates $r_j$ $(1 \leq j \leq N_i)$ of the clients of the multicast group $G_i$. First, a local server calculates $C_i$ for all $M$ multicast groups that it manages and finds the largest $C_m = \max_{1 \leq i \leq M} C_i$. If it exceeds a split threshold $T_s$, the local server divides the multicast group $m$ into two.

To find the appropriate splitting, we assume that clients are ordered in ascending order of reported rate $r_j$. Thus, the first client is CLR and the sending rate $R_m$ of group $G_m$ is equal to $r_1$. The local server defines the cutting point $b$, which is the border between two multicast groups. We first set the cutting point between the lowest client and the second lowest, $b = 2$, and calculate the coefficient variations for both groups, i.e., client 1 and clients 2 through $N_m$, and derive their average. We vary the cutting point $b$ from 2 to $N_m$ and find the point that leads to the minimum average in all possible combinations. Finally, the multicast group $m$ is split into two at the determined cutting point. The local server assigns a new multicast address to a multicast group of clients $b$ through $N_m$ and sends a video stream adjusted to client $b$.

### B. *Merging Multicast Groups*

After splitting multicast groups, a local server tries to merge a pair of multicast groups that carry independent video streams of a similar

level of quality for efficient use of system resources. To avoid thrashing, those multicast groups that are split on the same control timing are excluded from sets of multicast groups to be merged. For a pair of $R_i$ and $R_{i+1}$ ($1 \leq i \leq M-3$), a variation coefficient $V_i$ is calculated to quantify the similarity. Here we assume that multicast groups are ordered in ascending order of the sending rate. The lowest variation coefficient is compared to the threshold $T_m$. If the coefficient is smaller than the threshold $T_m$, the corresponding pair of multicast groups is merged into one.

## IV. Management of Multicast Group between Local Servers

Consider the case that congestion occurs on a link that constitutes a multicast tree. Clients behind the congested link suffer from the congestion. Under this condition, they experience longer RTT and higher loss ratio. Furthermore, they report lower TCP-friendly rate values. Consequently, all clients in the multicast group are forced to receive and perceive a video stream of degraded quality. Knowing the diversity of reception conditions, a local server splits the multicast group into two. Those clients that are not affected by the congestion are isolated from the others and receive a video stream of an improved quality. However, the clients behind the congested link still suffer from a low-quality video stream. This implies that the group membership management on a single local server does not contribute to the improvement of reception conditions.

In this section, we propose moving a multicast group to another local server in the same stub network, expecting that a newly established multicast tree avoids the congested bottleneck link on the current multicast tree. Since a local server does not have the means to directly obtain information on the available bandwidth, network topology, and topology of multicast trees, it investigates the possibility of improving the reception conditions through a probe phase. The local server that intends to move a multicast group, called a current server hereafter, chooses a multicast group whose sending rate is stable and the lowest among all multicast groups that it manages. The reason is that the video quality forms a monotonically increasing convex function of the coding rate and the same additional bandwidth leads to a higher gain in the video quality for a session of the lower sending rate than that of the higher rate. Then it selects a candidate server among local servers in the same domain. The candidate server initiates video multicasting on the multicast group to move. Members of the multicast group simultaneously receive video streams from both the current and the candidate servers and send feedback messages to both servers. Based on the reported TCP-friendly rate, and the sending rates, the current server decides whether the multicast group should be moved.

### A. Signaling Mechanism

The current server communicates with a candidate server and clients as illustrated in Fig. 4. Detailed descriptions of signaling procedures are as follows. A current server sends a request (probe_req) to a candidate server to initiate a probe phase. The candidate server receiving the request decides whether to accept it according to the number of multicast groups that it manages and the load. In addition, if it has already been engaged in other probing, where it is the current server or the candidate server, it rejects the request. When the candidate server accepts the request, it sends back an acceptance

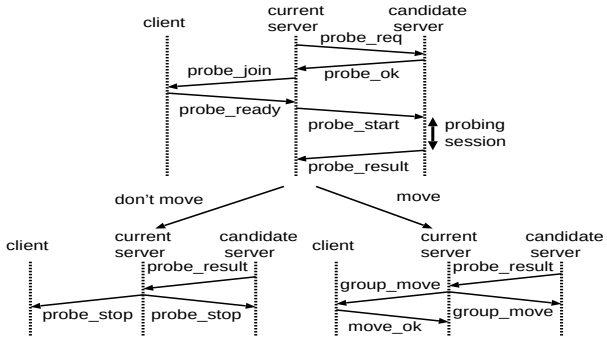


Figure 4: Signaling in move

message (probe_ok) to the current server and prepares a video stream. On receiving the acceptance message, the current server sends messages (probe_join) to all clients of the multicast group to inform them of the multicast address for probing. A client that receives the message joins the probing multicast group specified by the current server and returns a ready message (probe_ready). The current server requests the candidate server to initiate video multicasting by sending a message (probe_start) that indicates the probing multicast address. Then, the candidate server sends a video stream to the probing multicast group and sets a probing timer. In this paper, we empirically set a probing timer to 1.5 seconds, which is three times as long as the initial maximum RTT in each multicast group. The sending rate of the probing session is controlled by TFMCC as in the other multicast groups.

When the sending rate of the probing session becomes stable or the probing timer expires, the candidate server sends a message (probe_result) to the current server to report the sending rate. The candidate server compares the sending rate to the reported rate. If it considers it worthwhile to move the multicast group, it delegates the management by sending a move message (group_move) to the candidate server and clients. The candidate server continues multicasting, and the clients stop receiving the video stream from the current server and send confirmation messages (group_ok) to the current server. Finally, the current server stops video multicasting. On the other hand, if the current server considers that the reception condition cannot be improved by a move, it sends stop messages (probe_stop) to the candidate server and clients. The candidate server then stops video multicasting and the clients leave the probing multicast group.

### B. Stability of Sending Rate

In the preceding subsections, we discussed the stability of the sending rate. In our scheme, the stability of the sending rate of multicast group $i$ is expressed by the coefficient variation $S_i = \frac{v_i}{a_i}$. Here, $v_i$ stands for the standard deviation of sending rates that the local server adopted and applied to the multicast group $i$ during the preceding reorganization interval, and $a_i$ is the average of them. If $S_i$ is higher than the threshold $T_v$, the multicast group $i$ is considered unstable.

### C. Decision Algorithm on Moving Multicast Group

A local server must decide whether to move a multicast group based on limited information. At the time of decision, the local server knows the sending rate of the current multicast group and that of the probing multicast group. If multicast trees of those groups are com-

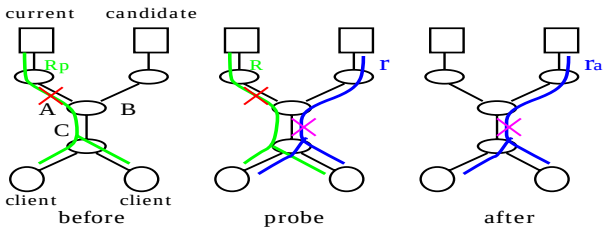


Figure 5: Example of successful move

pletely independent and there is no other session in the network, the multicast group whose sending rate is higher than the other apparently provides clients with a video stream of a higher quality. However, in an actual situation, traffic on a multicast group directly and indirectly affects the other multicast session. They might share the same link. Even if trees are separatedly established, there might be a TCP session that goes through both trees, thus bridging the influence of the rate control on one multicast tree to the other tree.

To clarify the problem, we categorize possible situations by combinations of three conditions: before probe, during probe, and after probe. For each condition, we then consider the locations of bottleneck links of each multicast tree. As a result, we derive sixteen patterns as possible situations. One simplified example is illustrated in Fig. 5. We define $R_p$ [bps] and $R$ [bps] as the sending rate from the current server before and during a probe phase, respectively. We also define $r$ [bps] and $r_a$ [bps] as the sending rate from the candidate server during and after the probe phase, respectively. Each of the three topologies in the figures corresponds to the condition before, during, and after the probe phase, respectively, from left to right. In each topology, the left square stands for the current server and the right one for the candidate server. Ellipses are intermediate multicast routers, and circles indicate clients. Curved lines originating from a server to two clients illustrate multicast trees. A cross on a tree represents a congested link or a bottleneck link. To make the explanation easier, three links are named A, B, and C as shown in the figures.

In Fig. 5, on a multicast tree from the current server to clients, link A has the least available bandwidth due to congestion or small link capacity before the probe phase. Thus, it dominates the sending rate of the current multicast tree. During the probe phase, the available bandwidth of link C is shared among two sessions. However, in this scenario, the bottleneck of the current tree still remains link A, while that of the candidate tree is link C. This occurs when the bandwidth available to video multicasting on link A is smaller than half that on link C and link B has plenty of bandwidth. When the current server terminates the session and moves the multicast group, the available bandwidth on link C is fully used by the new session, but it remains the bottleneck as we assumed. Let us consider the relationship among $R_p$, $R$, $r$, and $r_a$ in this scenario. $R_p$ is equal to $R$ because the bottleneck does not change on the current tree. During the probe phase, $R < r$ holds because the sending rate of the current tree is limited by the bottleneck link A, and the remaining bandwidth on link C is used by the probing session. Finally, since the current session is terminated, $r_a$ is larger than $r$. In this case, we obtain $R_p = R < r < r_a$, and consequently we can expect improvement in the reception conditions by moving the multicast group.

We investigated all sixteen situations to find such relationships among $R_p$, $R$, and $r$ that definitely lead to the improvement of reception conditions, i.e., $R_p < r$. Our conclusion is that it is worthwhile

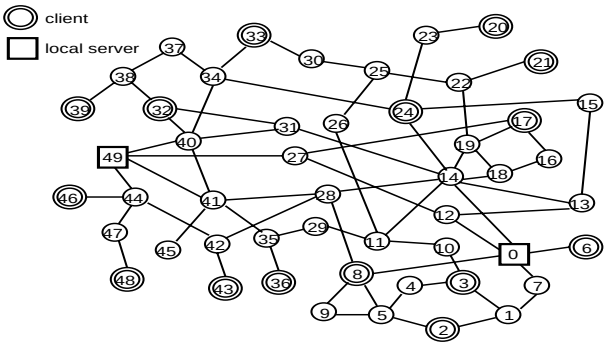


Figure 6: Simulation environment

to move a multicast group to another local server if the sending rate of the current group during the probe phase is higher than that of the probing group.

## V. Simulation Results

In this section, we show some simulation results of our approach. Figure 6 illustrates the topology of the network we used in simulation experiments. It is generated by a Waxman algorithm with parameters $\alpha = 0.14$, which represents the average node degree, and $\beta = 0.3$, which represents the ratio of the number of long edges to that of short edges. Each link has a capacity of 5 Mb/s and a propagation delay of 10 ms. Among fifty nodes, nodes 0 and 49 are local servers, which are indicated as square. Fifteen nodes have a client, and they are indicated as double circles. Initially, the local server 0 has eight clients, i.e., 2, 3, 6, 8, 17, 20, 21, and 24, and the local server 49 has the remaining seven clients. Both local servers start video multicasting with a single multicast group at time 0 second and complete it at 600 seconds. Throughout the video sessions, each node randomly initiates a TCP session to a randomly chosen node. The time that a TCP session lasts follows an exponential distribution whose average is 100 seconds, and the time between the end of a session to the next session on a node follows an exponential distribution whose average is 200 seconds. From 150 seconds to 400 seconds, in addition to the above mentioned sessions, each node from 0 to 12 randomly initiates a TCP session to a node randomly chosen from nodes 0 to 12 to intentionally cause congestion. Both session duration and inter-session time follow an exponential distribution whose average is 100 seconds. The thresholds for splitting $T_s$ and merging $T_m$ are both 0.2 and $T_v$, which is the threshold for stability of the sending rate at 0.25, and the re-organization interval is 10 sec.

Figure 7 illustrates the variation in the sending rate of local servers without re-organization from 100 to 450 seconds. When the network is congested from 150 seconds to 400 seconds, only three clients, i.e., 2, 3, and 8, out of eight are affected in the multicast group of local server 0. However, the other five clients also suffer from the congestion since they are in the same multicast group throughout the simulation experiment.

When we conduct splitting and merging, multicast groups are dynamically re-organized in accordance with network conditions on local server 0 as shown in Fig. 8. Each line corresponds to a multicast group. From 160 seconds, the local server 0 separates clients 6, 17, 20, 21, and 24 from those affected by congestion by splitting the initial multicast group. As a result, those clients can receive a

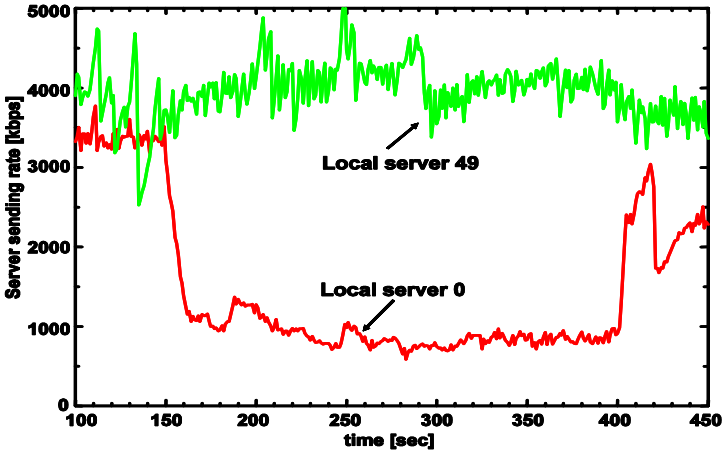


Figure 7: Sending rate variation without re-organization

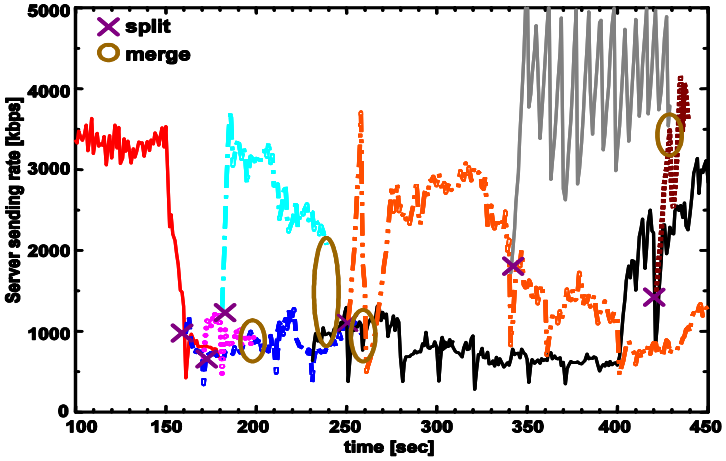


Figure 8: Sending rate variation of local server 0 with split/merge

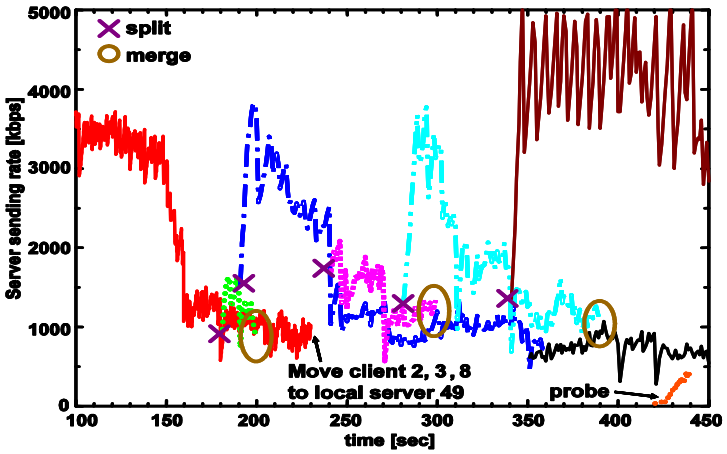


Figure 9: Sending rate variation of local server 0 with split/merge/move

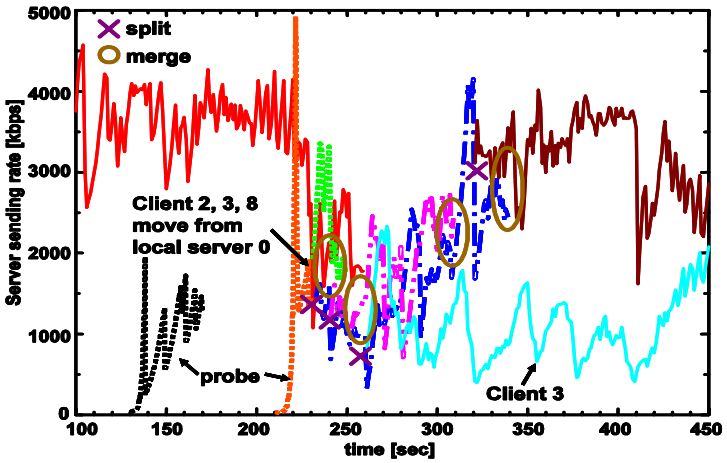


Figure 10: Sending rate variation of local server 49 with split/merge/move

video stream of an improved quality from 180 seconds. On the other hand, three clients continue to be affected by the congestion, and the sending rate of their multicast group is similar to that in Fig. 7. We observe that unintended and occasional re-organizations occur. For example, at 240 seconds, those split multicast groups are merged into one again. This is because additional TCP sessions cause fluctuation in the sending rates of both multicast groups.

Finally, we show results of the case where inter-server re-organization is carried out in Figs. 9 and 10. In Fig. 9, the multicast group affected by the congestion is moved to the local server 49 at 230 seconds. The moved multicast group introduces fluctuations of reception conditions on multicast trees from the local server 49 and then several re-organizations occur as shown in Fig 10. Finally, client 3, which cannot avoid congestion by re-construction of a multicast tree, organizes a dedicated multicast group. The other nine clients that are free from the congestion are accommodated in another multicast group and receive a video stream of a higher level of quality.

To summarize the section, we conclude that our re-organization scheme dynamically manages multicast groups and provides clients with video streams of a quality corresponding to the network conditions that individuals experience.

## VI. Conclusions and Future Work

In this paper, we proposed a method to effectively delegate the management of a multicast group between local servers in the same domain to improve the reception conditions that are degraded by congestion. Combined with splitting and merging, a video distribution service that takes into account client-to-client heterogeneity can be provided to users. To avoid the unintended and unnecessary re-organization observed in the simulation experiments, future research should include a way to carefully choose threshold values and other parameters.

## References

[1] Steven McCanne, Van Jacobson and Martin Vetterli, "Receiver-driven layered multicast," in *ACM SIGCOMM*, vol. 26,4, pp. 117–130, August 1996.

[2] Héctor Akamine, Naoki Wakamiya, and Hideo Miyahara, "Congestion-adaptive video multicast in an active network," in *Technical Report of the IEICE(NS 2002-59)*, pp. 27–30, June 2002.

[3] S.F.Bush and A.B.Kulkarni, *Active Networks and Active Network Management*. Kluwer Academic/Plenum Publishers, 2001.

[4] Jörg Widmer and Mark Handley, "Extending equation-based congestion control to multicast applications," in *Proceedings of SIGCOMM '01*, August 2001.

[5] Jörg Widmer and Mark Handley, "TCP-friendly multicast congestion control (TFMCC): protocol specification," tech. rep., IETF Internet Draft (draft-ietf-rmt-bb-tfmcc-01.txt), November 2002.