Cooperative Inter-stream Rate Control Scheme for Layered Multicast

Masato KAWADA Hiroyuki MORIKAWA Tomonori AYOYAMA
School of Engineering, The University of Tokyo
{kawada, mori, aoyama}@mlab.t.u-tokyo.ac.jp

Abstract

In a heterogeneous network, layered multicast is useful which enables receivers to determine the rate of a stream adaptively to their various capacities. Previous rate control schemes for layered multicast, however, treat each stream independently and cannot allocate the requested bandwidth to each stream when receivers subscribe to multiple streams. In this paper, we propose an inter-stream rate control mechanism for layered multicast called Multiple Streams Controller (MSC), which is located at each receiver and provides cooperative rate control among layered streams by end-to-end. In our method, MSC infers the tree information shared among streams and learns the detail network state by analyzing packet loss of each stream. Judging from the information, MSC can control the rate of streams cooperatively when congestion occurs or the receiver requests the bandwidth re-allocation of streams. We give some simulation results and confirm that MSC can allocate requested bandwidth by releasing the layer of other streams and provide the stability of the quality with less packet losses and oscillations.

1. Introduction

Recently the Internet is becoming the next network infrastructure, on which the multimedia data such as audio and video are expected to traverse in the near future. Multicast networking[1] is the effective method for servers to deliver multimedia services to a lot of subscribers, and is poised to become important enablers of new applications such as internet TV and network collaboration.

Multicast, however, delivers data at a single rate to several receivers in a heterogeneous network, where receivers differ in processing capabilities, available bandwidth and network connectivity. A single rate stream cannot satisfy both receivers on low bit-rate network and ones on high bit-rate network. To solve that problem layered multicast is a viable solution for providing an adaptive rate control in a heterogeneous network. In layered multicast, a stream is encoded into multiple layers, and each receiver subscribes to as many layers as its network connection and processing capability allow for and decodes them[2]. Several techniques are proposed for a receiver to determine the subscription level (the number of receiving layers) [3, 4, 5, 6].

In this paper we assume a video conference session or multi-channel broadcasting service, where users receive multiple streaming media flows such as audio and video transmitted from multiple senders by multicast at the same time. When a user receives multiple streams, his preference for each stream is expected to be different and dynamically changeable. If his available bandwidth is limited and shared among streams in such a situation, it is desirable that the bandwidth is effectively allocated to streams adaptive to his preference because it can increase his utility of the entire session. In addition it is strongly desirable for decoders to receive data at stable transmission rate with less packet loss and delay because the quality of decoded data is adversely affected by them. Less bad influence can bring better performance for users.

Previouse rate control methods, however, treat each stream independently without regard to other streams. If streams pass the same bottleneck link and share the limited bandwidth (which are called sharing streams in the following), they may compete the resource each other without knowing they share the same resource. This may cause the unfairness of subscription level and congestion collapses[7]. Needless to say, flexible bandwidth allocation adaptive to receiver’s interest is impossible without consideration and cooperation among streams.

Our goal of this paper is to design an end-to-end inter-stream control scheme for layered multicast which enables to adjust the allocated bandwidth of sharing streams according to the predefined policy. In this paper we use Receiver-driven Layered Multicast (RLM)[3] as basic method for receiving layered multicast streams, and establish Multiple Streams Controller (MSC) mechanism over RLM at each receiver. MSC can control RLM operations taking into account all streams the receiver subscribes to. MSC has two main schemes. One is to infer the sharing streams and the sharing receivers by investigating the relationship
of packet losses in congestion. The sharing receivers are the receivers whose bottleneck link is the same link. MSC judges whether it can change the bandwidth allocation among streams by that sharing information. Another is to control the bandwidth of streams according to both sharing information and receiver’s requests. MSC changes the bandwidth allocation among sharing streams by join/leave layers. We also show by simulation that MSC not only achieves flexible inter-stream rate control among sharing streams but also keeps the entire loss rate and the quality oscillation lower than those without MSC.

This paper is structured as follows. In Section 2 we describe the problems of RLM and the basic mechanism of our algorithm. Section 3 describes the detailed algorithm. In Section 4 we analyze and evaluate its behavior by simulation. Finally, Section 5 concludes this paper.

2. Approach

2.1. Related works and their problems

There are some techniques of a receiver-driven rate adaptation for layered multicast which don’t require any new mechanisms within the network and are transparent to senders: by using experimental join/leave at each receiver[3] or some agent receivers[4], the equation between throughput and loss rate like TCP[5], bandwidth estimation results of packet pair[6]. Receiver-driven rate control methods have two main mechanisms: the probe of empty bandwidth with trial joining to the next layer and loss detection, and the rate degradation on congestion by dropping the extended layer to reduce the congestion. These two mechanisms realize the similar rate control as that of TCP.

In RLM[3], each receiver determines its subscription level (the number of layers) by performing what are called join-experiments. A join-experiment is that a receiver subscribes to an additional layer and observes if the subscription degrades its reception quality as opposed to improving it. If congestion occurs due to the experiment in a certain time (called detection-time), the receiver immediately drops the layer and doubles the join-timer which is the interval to the next join-experiment. If not, the experiment is successful and the receiver does a join-experiment to next layer. And if packet loss rate measured in a detection-time exceeds a configured threshold when the stream doesn’t conduct join-experiment, the receiver regards the congestion as continuous one and drops the subscription level. Moreover, a receiver multicasts a join-experiment report identifying the experimental layer to other receivers when doing the experiment so that all receivers can learn the network condition from failed experiments.

RLM is useful for the heterogeneous network where only one layered multicast stream is transmitted into the link with various capacities. If a receiver subscribes to more than two layered streams which pass the same bottleneck link and share its limited bandwidth, however, each stream controls its rate with no regard to other streams in the current model (refer to left side of Fig.1). Therefore there occurs the following problems: (1) receivers can’t adjust the allocated subscription level among streams, (2) if there are streams occupying the bottleneck link, the new stream passing the same link has little available bandwidth, which causes the unfairness of allocated bandwidth among streams, (3) if join-experiments are overlapped among streams, the experiments are likely to interfere with each other and be failed (4) if temporary congestions caused by join-experiments of other streams happen continually, a receiver misrepresents this congestion as continuous one and drops the highest layer.

2.2. Overview of inter-stream control

To realize the inter-stream rate control that the legacy model cannot provide, we establish a mechanism over layered streams called Multiple Streams Controller (MSC) on
each receiver (refer to the right side of Fig.1). In the following, we focus on RLM[3] as the receiver-driven rate control method, which is major rate control protocol in layered multicast and implemented in ns[12]. As other rate control methods in layered multicast also determine the number of layers by detecting packet losses on trial joining to the next layer, we believe MSC is applicable to other layered multicast protocols.

In our method, we extend RLM to RLM++ which has an interface to MSC, through which RLM++ and MSC exchanges control messages such as packet loss, join-experiment report, and instruction(join/leave). RLM++ must obey the instruction of MSC though the other function is similar as RLM.

Fig.2 shows the architecture of MSC. The main work of MSC is to control all receiving streams cooperatively with regard to each other. For that purpose MSC must first observe the condition of RLM++ streams and infer the information on the bottleneck link such as sharing streams and sharing receivers by gathering and analyzing the messages from all RLM++ streams through the interface to RLM++ streams. Each MSC has its congestion table for recording the congestion result. When a join-experiment of each stream is conducted MSC records its consequent congestion of all receiving RLM++ streams in its congestion table. MSC can infer the sharing streams from the fact that they experience similar effect of congestion such as packet loss and delay, and sharing receivers from the fact that they influence the congestion by their join-experiment each other.

Based on the information of the bottleneck link, MSC controls join/leave of RLM++ streams. The control algorithm is devised into three main functions in Fig.2. RLM++ operation supporter instructs join-experiment and leave to RLM++ streams considering other streams’ condition in order to achieve stable quality of service without bandwidth competition. Base layer protection guarantees the minimum quality of streams by protecting the reception of the base layer. And adjustment of subscription level achieves a flexible inter-stream rate control among sharing streams according to predefined policy such as the receiver's interest and subscription level fairness.

3. Cooperative inter-stream rate control algorithm

In this section we explain inter-stream control functions with MSC and RLM++ referring to Fig.2 in detail. In the following, we use an expression $L_{X,2}$ as the second layer of stream $X$ for the sake of convenience.

The operations of RLM++ different from RLM are itemized as follows:

3.1. RLM++ operation support

Before explaining inter-stream algorithm, we introduce extended RLM called RLM++ to cooperate with MSC. RLM++ has an interface to MSC in order to exchange messages, through which RLM++ informs MSC of join-experiment report and packet loss and receives instruction from MSC. RLM++ asks MSC about join/leave and obeys the instruction from MSC. Moreover RLM++ adds several items to join-experiment report, [the receiver conducting the join-experiment, the experimental stream, its subscription level]. Basic operation of RLM++ such as timers, parameters and states is the same as original RLM.

MSC observes the condition of all RLM++ streams by receiving their messages. When RLM++ asks to MSC whether he should join/leave MSC instructs with regard to other streams.

Fig.3 shows the join-experiment algorithm of RLM++ with MSC. When the join-timer ($T_{J}$) for next join-experiment expires, RLM++ checks whether the state of the RLM++ stream is Stable [in which there is no packet loss, no ongoing join-experiments of other receivers to lower-level layer] at first. If the RLM++ stream satisfies the condition,
RLM++ asks the experiment to MSC and then MSC checks the same point of other receiving streams. If the condition is also satisfied MSC instructs the join-experiment to the RLM++ stream, otherwise RLM++ and MSC checks the conditions again after a short-time ($T_W$). By this method MSC prevents overlaps of join-experiments among receiving streams.

When one of receiving streams runs join-experiment and examines congestion, MSC also examines congestion of all receiving streams. If a stream detects congestion while MSC is examining congestion, MSC judges that the other stream’s join-experiment has caused the congestion, and instructs the stream to wait for a detection-time until the congestion finishes. If a stream detects congestion but MSC isn’t examining congestion, MSC judges that the congestion is unrelated to receiving streams and instructs the stream to investigate the packet loss rate. If the loss rate exceeds a threshold, RLM++ considers the congestion as continuous one and decreases the subscription level. In this way, MSC specifies the cause of the congestion by observing all receiving streams, and copes with the congestion based on the cause among streams. MSC can prevent the wasteful decreasing of the subscription level happening by misrepresenting congestion due to other streams’ join-experiments as continuous congestion.

### 3.2. Congestion table

Congestion table has the congestion results caused by join-experiments, which consist of the experimental stream, the receiver conducting the experiment, the subscription level of the receiver and the list of streams which detect congestion caused by the experiment. Each MSC at a receiver has its own congestion table, and records the congestion results from all receiving streams in its table. By analyzing this table, MSC can grasp the sharing condition of bandwidth in a bottleneck link among streams and judge whether an inter-stream adjustment of the subscription level is possible.

First, we describe how to construct a congestion table using Fig.4 as an example. The topology consists of two senders ($S_X$, $S_Y$) and two receivers ($R_1$, $R_2$). Link 3-4 and 3-5 in black are bottleneck links. The value in square and circle shows the number of layers of Stream $X$ ($St_X$) and Stream $Y$ ($St_Y$) passing the link, respectively. In the table of Fig.4, (host) is the receiver who conducted the join-experiment, (layer) is the subscription level of the receiver, (cong) is a list of congested streams by the experiment.

We assume the case that $R_1$ increases its subscription level of $St_X$ from 3 to 4. First, $R_1$ multicasts the join-experiment report including the information [$R_1$, $X$, $4$] to other receivers and adds a layer. Upon reception of the report, $R_2$ checks the subscription level in the report with that of $R_2$. If the subscription level in the report is bigger than that of $R_2$, it is possible that the experiment affects the streams $R_2$ receives. Hence, MSC of $R_2$ and MSC of $R_1$ who conducts the experiment start to detect congestion on all receiving streams. When the join-experiment is conducted in the state of Fig.4, congestion occurs and random packet losses happen on $St_X$ and $St_Y$ at bottleneck link 3-4, then $R_1$ and $R_2$ at downstream of link 3-4 detect the losses. MSC records the result of the congestion to its own congestion table. In this experiment, ($R_2$ (the receiver conducting the experiment), 3 (the subscription level of the receiver), $XY$ (the stream which detected packet losses)) is recorded at the part Stream $X$ in the tables of $R_1$ and $R_2$.

And if a receiver decreases its subscription level it multi-
casts a report with the level, and other receivers update their congestion table according to the level in the report.

Next, we describe how to analyze congestion table by using the congestion table at \( R_1 \) in Fig.4. If there is other receiver which has the same subscription level as \( R_1 \) (by referring at (layer)) in the table, it shows that \( R_1 \) and the other receiver are sharing the common bottleneck link. On the other hand, if the subscription level of other stream is different from \( R_1 \), it shows that \( R_1 \) and the other receiver have different bottleneck link each other. It’s because \( R_1 \) can detect the congestion that occurs at any path from the source to \( R_1 \), where there is only one bottleneck link of specified number of layers. In table \( R_1, R_1 \) and \( R_2 \) are sharing the common bottleneck link on \( S_{TX} \) because \( R_1 \) and \( R_2 \) have the same subscription level 3. But on \( S_{TY} \) the bottleneck link of \( R_1 \) is different from \( R_2 \) because \( R_1 \) has the different subscription level 1 from \( R_2 \).

And if there are other streams in the stream list of congested streams (by referring at (cong)), it shows that the other streams pass through the same bottleneck link and share bandwidth of the link among them. Since there is \( Y \) at (cong) in the Stream X part of the table, \( S_{TY} \) passes through the bottleneck link of \( S_{TX} \).

In this way, MSC can grasp the sharing state of bandwidth in a bottleneck link both among receivers and among streams from congestion table. The information in the table becomes the clue of performing the adjustment of subscription level among streams.

### 3.3. Base layer protection

It is desirable that receivers can at least subscribe to the base layer of RLM stream without congestion. To receive a base layer of a new RLM++ stream without congestion when other stable streams occupy almost all available bandwidth and there remains least bandwidth for the new stream, MSC and RLM++ should guarantee the bandwidth for the base layer by degrading the subscription level of the other stream.

When a receiver subscribes to the base layer of a new RLM++ stream, it multicasts join-experiment report to other receivers and examines congestion in all receiving streams. If the new stream detects congestion, MSC decides the dropping stream among the other streams which detect congestion. If some receivers should decrease their subscription level of the dropping stream to give bandwidth to the base layer, MSC multicasts the request of dropping and the receivers who detect congestion caused by the base layer decrease the subscription level when they receive the request. If \( R_2 \) starts new \( S_{TX} \) in the state of Fig.4, for example, \( R_1 \) and \( R_2 \) detects congestion occurring at link 3-4 on all streams. \( R_2 \) decides the dropping stream, \( S_{TX} \) or \( S_{TY} \), and multicasts a drop request. Consequently, \( R_1 \) and \( R_2 \) decrease the level of the dropping streams.

### 3.4. Adjustment of subscription level

When multiple streams are sharing bandwidth of bottleneck link, a stream (called joining stream) can increase the subscription level if one of other sharing streams (called dropping stream) decrease its subscription level to provide adequate bandwidth for the next layer of the joining stream. MSC can find the situation in which MSC can adjust the subscription level among streams by analyzing congestion table.

First, MSC finds a dropping stream from the list of congested streams, i.e. (cong). MSC can find that \( S_{TX} \) and \( S_{TY} \) are sharing streams from the congestion table at \( R_1 \) in Fig.4. In this case MSC should drop the subscription level of \( S_{TX} \) if wants to increase the subscription level of \( S_{TY} \). Next, MSC searches other receivers that have the same subscription level of the dropping stream in the congestion table in order to drop the maximum layer from the bottleneck link and get bandwidth for the joining stream. If there are the receivers MSC must drop the layer with them, and if no receivers MSC had better drop the layer by itself. In Fig.4, if \( R_1 \) wants to increase the level of \( S_{TX} \), MSC had better drop the maximum layer of \( S_{TY} \) by itself because \( R_1 \)'s congestion table shows that \( S_{TY} \)'s bottleneck link of \( R_1 \) is different from the one of \( R_2 \). However, if \( R_1 \) wants to increase the level of \( S_{TY} \) MSC must cooperate with \( R_2 \) in dropping the maximum layer of \( S_{TX} \) because \( R_1 \)'s congestion table shows that the \( S_{TX} \)'s bottleneck link of \( R_1 \) is same as \( R_2 \).

If the cooperation is needed the MSCs must negotiate with them whether they adjust the layers judging from their interest in the streams. (We doesn’t touch on it in this paper, but it is very important and future works.)

MSC can adjust the subscription level among streams according to receiver’s request or a predefined rule.

### 4. Simulation and evaluation

In this section we present simulation results of simple topology to evaluate the operations. We implemented the MSC and RLM++ in network simulator ns[12]. We compare the results in receiving multiple RLM++ streams cooperatively with MSC, and those in receiving multiple RLM streams independently.

#### 4.1. Simulation model

Senders are hierarchical CBR sources, which send 4 layers whose bandwidth are 100kbps, 100kbps, 200kbps, 400kbps respectively[13]. Each receiver runs only RLM or RLM++ with MSC when it receives multiple layered streams.
Table 1. Simulation parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>link delay</td>
<td>200 [ms]</td>
</tr>
<tr>
<td>packet size</td>
<td>1000 [byte]</td>
</tr>
<tr>
<td>queue size</td>
<td>10 [packet]</td>
</tr>
<tr>
<td>join-timer (init.)</td>
<td>10 [sec]</td>
</tr>
<tr>
<td>join-timer (max)</td>
<td>100 [sec]</td>
</tr>
<tr>
<td>detection-timer (init.)</td>
<td>5 [sec]</td>
</tr>
</tbody>
</table>

Tab.1 shows the common parameters for RLM algorithm in our simulations, and the two simulation topologies are illustrated in Fig.5. Topology (1) consists of three sources ($S_X, S_Y, S_Z$), two receivers ($R_1, R_2$) and two bottleneck links (4-5,5-6). The bandwidth B of link 4-5 is 650kbps and the bandwidth B' of link 5-6 is 350kbps. The bandwidth of other links is 1000kbps, which is rich enough not to cause congestion. $R_1$ starts receiving $St_X$ at 5[sec] and $St_Y$ at 40[sec]. At 60[sec] the subscription level of them becomes $(X, Y) = (2, 1)$, where $R_1$ consumes 300kbps and the bottleneck link of $R_1$ is link 5-6. $R_2$ starts receiving $St_X$ and $St_Y$ at 5[sec]. After, at 60[sec] the subscription level of them becomes $(X, Y) = (2, 3)$, where $R_2$ consumes 600kbps and the bottleneck link of $R_2$ is link 4-5. Here, we confirm the operation of base layer protection and adjustment of the subscription level by observing the bandwidth re-allocation of streams at a bottleneck link.

Topology (2) consists of three sources and multiple receivers separated by a bottleneck link 4-5. Here, we explore the scalability of the performance with respect to receiver size.

4.2. Results and evaluation

In this section, we present the results of simulations on the two topologies described above.

4.2.1. Base layer protection

We confirm the operation of base layer protection by observing the bandwidth of layered streams at bottleneck link 4-5 in topology (1). $R_2$ joins $St_Z$ at 100[sec] when $St_X$ and $St_Y$ has occupied the bandwidth of the bottleneck link 4-5 and there isn’t enough bandwidth for the base layer of $St_Z$ to pass through the bottleneck link. In this case, congestion occurs at the bottleneck link and all three streams passing the link detect packet losses.

Fig.6 shows the bandwidth allocation when receivers run RLM to receive each stream independently with no consideration to other streams. Needless to say, no base layer protection is conducted. After $St_Z$ joins at 100[sec], there has been congested for about 40[sec]. But $St_Z$ drops the base layer at 140[sec] because $St_X$ and $St_Y$ don’t drop their highest layer ($L_{X,2}$ or $L_{Y,3}$) and there is no bandwidth enough for the base layer of $St_Z$.

Fig.7 shows the bandwidth allocation when receivers run RLM++ and MSC to receive streams. In this case, when MSC recognizes a new stream $St_Z$ and detects the congestion caused by the join of $L_{Z,1}$, MSC instructs to drop the highest layer $L_{Y,3}$ at 104[sec] instead of dropping $L_{Z,1}$. As shown in Fig.7, the base layer protection algorithm enables $St_Z$ to gain enough bandwidth for its base layer and the three streams including $St_Z$ to transmit through the bottle-
neck link without packet loss after then.

4.2.2. Adjustment of subscription level

We confirm the operation of adjustment of subscription level by observing the bandwidth of each stream at bottleneck link 5-6 in topology (1). Fig.8 shows the congestion table at R1. Here, we find the situation in the shade part of the figure that R1 can adjust its subscription level by itself from the analysis of Section 3.4. Fig.9 shows that the subscription level changes from \((X, Y) = (2, 1)\) to \((X, Y) = (1, 2)\) at 160[sec]. At that time, MSC at R1 judges the adjustment from both the information of congestion table (Fig.8) and the receiver's demand : for example, \(R_1\)'s preference for \(St_Y\). Then MSC instructs \(St_X\) to drop the layer \(L_{X,2}\) and \(St_Y\) to join the layer \(L_{Y,2}\) respectively.

4.2.3. Scalability

We evaluate the scalability of our algorithm with MSC and RLM++ in receiving multiple streams by varying the receiver size in topology (2). The three senders in topology (2) starts their own session randomly during the first 5[sec] of simulation time, and receivers start receiving each streams at random time between 5[sec] and 7[sec]. We investigate the three metrics which reflect the perceived quality of a real-time, loss-tolerant multimedia stream at the receiver : convergence time, loss rate, and subscription level stability.

Now we repeat the simulation for 5 times and take an average value over the 5 tests in order to reduce the effect of specific test.

Convergence time: Convergence time for a receiver is the time between when it starts and when it reaches its stable subscription level, and shorter convergence time is desirable. In this simulation model we define convergence time as the average time it takes for a receiver to receive two layers of each stream (X, Y, and Z). Fig.10 plots the convergence time versus receiver size. It illustrates that convergence time does not increase linearly with receiver size in both RLM and RLM++. It also illustrates that the convergence time of RLM++ is 5-10 second longer than that of RLM. This is because MSC don't accept the join-experiment during detection-time after the join-experiment of a base layer for base layer protection. However, we con-
sider the delay is short enough compared with the long period of receiving streaming data.

**Loss rate:** We use maximum loss rate as an evaluation metric, which is the maximum loss rate computed within sliding measurement window (1, 10, or 100 seconds). The short-term loss rate (window of 1 sec) represents the degree of congestion of a join-experiment, and the long-term loss rate (window of 100 sec) represents the frequency of congestion, namely, the frequency of failed join-experiments. Fig.11 plots the maximum loss rate versus receiver size. It illustrates that both RLM and RLM++ causes a little increase of loss rate when the increase of receiver size. It also shows that RLM++ realizes lower loss rate than RLM especially in the long-term loss rate because MSC prevents the overlap of join-experiment among receiving streams so there is less congestion by failed join-experiment due to join-experiment overlapping.

**Table 2. The average frequency of missdropping layers**

<table>
<thead>
<tr>
<th>receiver size</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLM</td>
<td>0.8</td>
<td>2.4</td>
<td>3.8</td>
<td>9.4</td>
<td>20</td>
</tr>
<tr>
<td>(per receivers)</td>
<td>0.16</td>
<td>0.24</td>
<td>0.19</td>
<td>0.32</td>
<td>0.4</td>
</tr>
<tr>
<td>RLM++</td>
<td>0.2</td>
<td>0.8</td>
<td>0.6</td>
<td>0.6</td>
<td>0.4</td>
</tr>
<tr>
<td>(per receivers)</td>
<td>0.04</td>
<td>0.04</td>
<td>0.02</td>
<td>0.02</td>
<td>0.008</td>
</tr>
</tbody>
</table>

**Stability of subscription level:** We use the frequency of misdropping layers as evaluation values of subscription level stability. Table 2 illustrates the scalability of the frequency of misdropping layers between RLM and RLM++. As the receiver size increases, the frequency of misdropping grows exponentially by RLM because the more receivers, the more collisions of join-experiments occur. But by MSC and RLM++ the frequency of misdropping keeps low because MSC can prevent collisions of join-experiments among receivers and among streams, and can distinguish the cause of congestion.

**5. Conclusion**

We have proposed the end-to-end inter-stream control method for layered multicast when each user receives multiple streams. We showed that Multiple Streams Controller (MSC) and RLM++ can do the base layer protection and the adjustment of subscription level among streams by constructing and analyzing congestion table. We evaluated the performance of MSC and RLM++ through simulation and showed that it achieves stable subscription level and well scaling when many users receive multiple RLM++ streams. If the control mechanism can be adopted to network collaboration, we think that effective bandwidth allocation among streams with user’s request and receiving streams with stable quality are possible.

**References**


