

Network-supported Rate Control Mechanism for Multicast Streaming Media

Kiyohide NAKAUCHI Hiroyuki MORIKAWA Tomonori AOYAMA
School of Engineering, The University of Tokyo
{nakauchi,mori,aoyama}@mlab.t.u-tokyo.ac.jp

Abstract

Layered Multicasting with layered source coding scheme enables efficient distribution of real-time streaming media over heterogeneous networks such as the Internet. In layered multicast, rate control is required to cope with congestion quickly because packet losses in a layer with higher priority lead greater degradation of received quality. In addition, rate control for layered multicast is required to assure fair bandwidth allocation among competitive layered multicast sessions at a congested link independent of their start time. In this paper, we present a novel rate control scheme for layered multicast, called Network-supported Layered Multicast (NLM). In NLM, routers distributed within the network support rate control. NLM avoids packet losses on a layer with high priority and achieves fair bandwidth allocation by filtering and signaling. A router filters a layer with low priority when it detects congestion, or when it is informed that the layer is needless by signaling. Simulation results show NLM offers significant advantages compared with receiver-driven rate control.

1. Introduction

Multicasting audio and video streams over the current Internet is an bandwidth-efficient approach for real-time multipoint applications. To utilize bandwidth effectively, these applications are required to cope with the heterogeneity of the Internet; difference of processing power of each receiver and of available bandwidth for each receiver.

Several approaches have been suggested to cope with the heterogeneity for real-time multicast applications over the Internet[1]. One approach is simulcast[2]. In simulcast, all the receivers are divided into several groups in accordance with their available bandwidth, and the sender transmits an original stream to each group with different sending rate. Another approach is locating video gateways within the network[3]. A video gateway transcodes an input video stream encoded using some scheme into a video stream encoded using another scheme with lower bandwidth, and for-

wards this stream down to the congested link which is a part of a multicast tree. More efficient approach is layered multicast with layered source coding[4, 5, 6, 7, 8, 9, 10]. In layered multicast, a sender of a session encodes an original stream into a base layer and several enhancement layers. A base layer has the highest priority and higher enhancement layer has lower priority. Then, a sender transmits each layer on a separate IP multicast group. Receivers join some groups from the base layer to adapt their receiving rate to their environments. In this paper, we refer to the mechanism of adapting the number of received layers as rate control.

Rate control for layered multicast must be designed to meet the following two requirements. First, rate control must prevent packet losses on a lower layer because they give more significant degradation of video quality. Second, rate control must assure fair bandwidth allocation among competitive sessions at a congested link. Note that fairness in this paper is defined as max-min fairness for layered multicast[6, 11].

In general, rate control for layered multicast is classified into two classes; receiver-driven scheme[4, 5, 6, 7, 8, 9] and network-supported scheme[10]. In receiver-driven rate control, each receiver makes decisions to add or drop an enhancement layer. On the other hand, in network-supported rate control, some of internal nodes such as routers or switches adopt priority dropping mechanism which drops lower priority packets belonging to a higher layer, and decide the number of received layers for each receiver.

Receiver-driven rate control has some limitations and cannot satisfy essentially the requirements shown above. Receiver-driven schemes assume that the network retains the uniform dropping mechanism such as drop-tail or Random Early Detection (RED)[12], where all data packets are treated equally with respect to drop precedence[4]. Although packet losses on a lower layer caused by random dropping lead to more significant degradation of received quality, receiver-driven schemes cannot but depend solely on packet losses to detect congestion because network has no mechanism to convey to receivers the explicit knowledge about the network conditions[13]. Inevitably this causes packet losses on a lower layer. Receiver-driven schemes

have another problem that receivers cannot respond to the congestion quickly because of both the delay for estimating network conditions and detecting congestion (estimation delay) and the delay for leaving a multicast group well known as IGMP leave delay[10]. This slow reaction to the congestion cause more packet losses on a lower layer. In addition, many works[4, 6, 14] show inability for receiver-driven schemes to achieve fairness among competing layered multicast sessions.

On the other hand, network-supported rate control with priority dropping can avoid packet losses on a lower layer because a router drops lower priority packets in the face of congestion on an outgoing link. But adjusting transmission rate only by priority dropping cannot achieve fairness because it is inevitable that some packets with low priority are sent needlessly, only to be dropped further downstream congested links. In order to assure fairness, a router needs to take account of congestion not only at directly downstream link but also at the other downstream ones on the path down to receivers.

As described above, traditional rate control schemes do not meet two requirements for layered multicast. In this paper, we present a novel network-supported rate control mechanism, called Network-supported Layered Multicast or NLM. In NLM, routers with rate control functions are distributedly located within the network. A router drops packets with low priority when it detects congestion. A router performs signaling to convey available bandwidth to upstream ones and also drops needless packets based on the available bandwidth of downstream links.

The remainder of this paper is organized as follows. In section 2 we show the basic network support mechanism. In section 3 we describe NLM architecture and present the network-supported rate control algorithm in details. Section 4 contains the results of a set of simulations, and conclude in section 5.

2. Network Support for Layered Multicast

General network support mechanism for preventing packet losses on a lower layer is priority dropping at a router detecting congestion[10, 15]. This mechanism can also provide quick response to the congestion if it drops packets in advance before the buffer overflows such as RED.

But traditional network-supported rate control, which adjusts transmission rate of each layered multicast session by priority dropping, has two drawbacks. The first one is inability to achieve max-min fairness. According to the definition of max-min fairness, the excess bandwidth a session cannot use up because of the limitation on another link are allocated fairly among all other sessions. But, in a network-supported scheme, a router cannot know the limitation on another downstream link. In order to assure fairness, sig-

naling mechanism is necessary for routers to inform upstream ones in a multicast tree of available bandwidth at downstream links.

The second drawback is to transmit a fraction of a layer which is useless for receivers because generally in layered encoding scheme a layer is required to be received without any loss in order to be decoded. Transmitting this useless layer results in the waste of bandwidth on the path down to the receivers. Although priority dropping with the knowledge of the limitation can achieve max-min fairness, strict fair bandwidth allocation to each session causes receivers to receive the fraction of the highest enhancement layer among the layers received by receivers. In order to avoid transmitting the fraction, a router needs a filtering mechanism which drops all the packets belonging to the highest enhancement layer, which has the lowest priority.

In summary, the basic idea of network-supported rate control presented in this paper is that a router filters a layer with the lowest priority among sessions when it detects congestion or filters a needless layer when the LMR is informed by signaling that the layer is filtered further downstream. This mechanism can prevent packet losses on a lower layer, and also can assure fairness among sessions.

3. NLM Architecture

In this section, we show the architecture of NLM. NLM works within the existing IP model and requires current IP multicast to add no new machinery. NLM is deployed to the network where only a best effort service is provided and routers adopt drop-tail.

3.1. System Model

NLM system consists of a sender, receivers, and routers with rate control mechanisms. A sender hierarchically encodes sending data of a session to several CBR streams, and sends all streams on a single multicast group. As routers and receivers cannot identify a layer from its multicast address in this model, the number indicating a layer is filled in somewhere in a packet header. A receiver joins a single multicast session on which all the layers of the session are transmitted.

In NLM, routers with rate control mechanisms of filtering and signaling are introduced within a network. We call this router as Layered-Multicast-capable Router or LMR. NLM does not require all the multicast-capable routers within a network to implement rate control mechanism. Ideally, LMRs are strategically located at boundaries between the areas with rich bandwidth and those with poor bandwidth, or at the points where congestion frequently occurs.

MGroup	Source	Session ID S_i	PrevID	Oif I_j	S_i	$L_{max}(S_i)$	$L_{cur}(I_j, S_i)$
S1	Src1	1	LMR1	1	1	2	2
S2	Src2	2	LMR2	2	2	3	3
S3	Src3	3	LMR3	3	3	4	4
S1	Src4	4	LMR4	2	1	2	2
...
...
...

Figure 1. Session State Table

3.2. Overview of Network-supported Rate Control

A LMR possesses a session state table shown in Figure 1 and performs rate control based on this table. A session state table stores two values for each output interface (oif); $L_{cur}(I_j, S_i)$ which shows the number of layers allowed to be transmitted for session S_i at oif I_j , and $L_{max}(S_i)$ ($= \max_j L_{cur}(I_j, S_i)$) which shows the maximum number of layers of session S_i the LMR can transmit. Note that a session is referred to the unique pair of sender address (Source) and multicast address (MGroup). The directly upstream LMR address (PrevID) is also stored for each session. PrevID is necessary for signaling because signaling packets are unicasted from a downstream LMR to upstream one.

Rate control for NLM consists of the following three key functions.

- **filtering** : A LMR decides which layer to filter based on the current bandwidth allocation or on the request from downstream LMRs.
- **signaling** : A LMR requests upstream ones to filter a needless layer or to transmit a layer without filtering.
- **session control** : LMRs keep consistency of session state tables with one another.

First, a LMR performs filtering at each oif independently. At I_j , a packet in S_i is filtered if the layer of the packet is larger than $L_{cur}(I_j, S_i)$. Therefore incrementing or decrementing $L_{cur}(I_j, S_i)$ enables rate adaptation of S_i at I_j . A LMR monitors the queue length and calculates the average queue length ($qlen^j$) at each oif I_j . A LMR detects congestion when $qlen^j$ exceeds the upper threshold ($qmax$), and on the other hand, it detects the congestion solved when $qlen^j$ is below the lower threshold ($qmin$) at I_j .

When a LMR detects congestion at I_j , the LMR selects the session S_{i_0} which has maximum L_{cur} among all S_i at I_j and decrements $L_{cur}(I_j, S_{i_0})$. Thus the LMR starts filtering the layer with the lowest priority among all the transmitted layers at I_j . On the other hand, when a LMR detects congestion solved at I_j , the LMR selects the session

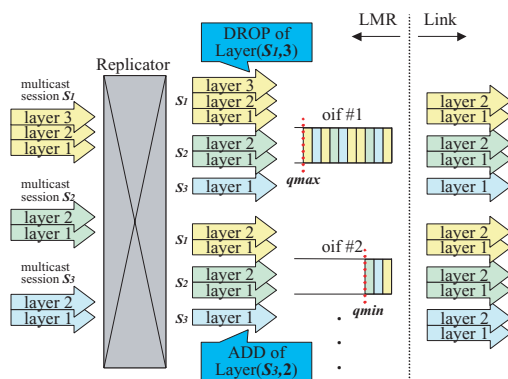


Figure 2. Overview of Filtering Mechanism

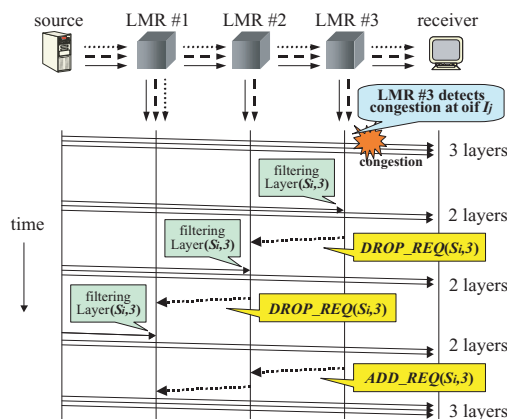


Figure 3. Overview of Signaling Protocol

S_{i_1} which has minimum L_{cur} among all S_i at I_j and increments $L_{cur}(I_j, S_{i_1})$. In this paper, we call the former mechanism as DROP and the latter as ADD respectively. Figure 2 shows the overview of filtering mechanism. In this figure, we express the L -th layer of S_i as $Layer(S_i, L)$.

Second, a LMR performs signaling after conducting DROP or ADD, if necessary. When a LMR conducts DROP and starts filtering $Layer(S_i, L)$ at I_j , the LMR sends $DROP_REQ(S_i, L)$ to the upstream LMR if $Layer(S_i, L)$ is no longer transmitted from any other oif and becomes needless for the LMR. The upstream LMR receiving $DROP_REQ(S_i, L)$ decreases $L_{cur}(I_j, S_i)$ to $L - 1$ at the oif where $DROP_REQ(S_i, L)$ has been received, and starts filtering $Layer(S_i, L)$. Then, if $Layer(S_i, L)$ also becomes needless for the LMR, it forwards $DROP_REQ(S_i, L)$ to the upstream LMR. Otherwise, $DROP_REQ(S_i, L)$ is discarded. $ADD_REQ(S_i, L)$ is also processed in the same way. Figure 3 shows the overview of signaling protocol.

Table 1. Parameters for Filtering Algorithm

add_intvl^j	time interval for the next ADD at I_j
add_intvl_min	minimum value of add_intvl^j
add_intvl_max	maximum value of add_intvl^j
add_time^j	last ADD time at I_j
$drop_intvl$	time interval for the next DROP
$drop_time^j$	last DROP time at I_j
$detect_period$	period for detecting congestion
$qlen^j$	average queue length at I_j
$qmax$	upper threshold to detect congestion
$qmin$	lower threshold to detect congestion
$qweight$	weight for calculating $qlen^j$ at I_j

Thus signaling enables each LMR to prevent transmitting needless layer. In this way, LMRs can assure fairness when signaling works together with filtering mechanism which allocates bandwidth fairly to each session at each oif.

Third, LMRs keep consistency of session state tables with one another by session control so that they can select the proper layer when they conduct DROP or ADD.

3.3. Filtering Algorithm

A LMR conducts rate adaptation by filtering when it detects congestion occur or been solved at an oif. Table 1 shows the parameters used for filtering algorithm. add_intvl^j , add_time^j , $drop_time^j$ and $qlen^j$ are variables and kept by each oif independently.

Before we show filtering algorithm, we explain the way to calculate $qlen^j$. We do not make use of the current queue length but the average one in order to prevent a LMR from decreasing transmission rate of a session when it receives temporal burst traffic and $qlen^j$ exceeds $qmax$ instantaneously. Whenever a LMR queues a packet at I_j , it calculates the current $qlen^j$ using both the current queue length (q_{cur}^j) and previous $qlen^j$ as follows[12]:

$$qlen^j = qweight \times q_{cur}^j + (1 - qweight) \times qlen^j$$

where $qweight$ is a weighted parameter which decides the changeability of $qlen^j$. We need to set $qweight$ carefully in accordance with the application based on its requirement and experimental results.

Figure 4 shows filtering algorithm. First, we explain DROP process. When a LMR detects congestion at I_j , it searches the target layer of DROP from a session state table. The LMR selects the target layer of DROP which meets the following two conditions:

1. $i_0 = i$, if $L_{cur}(I_j, S_i) = \max_k \{L_{cur}(I_j, S_k)\}$.
2. $L_{cur}(I_j, S_{i_0}) \geq 2$.

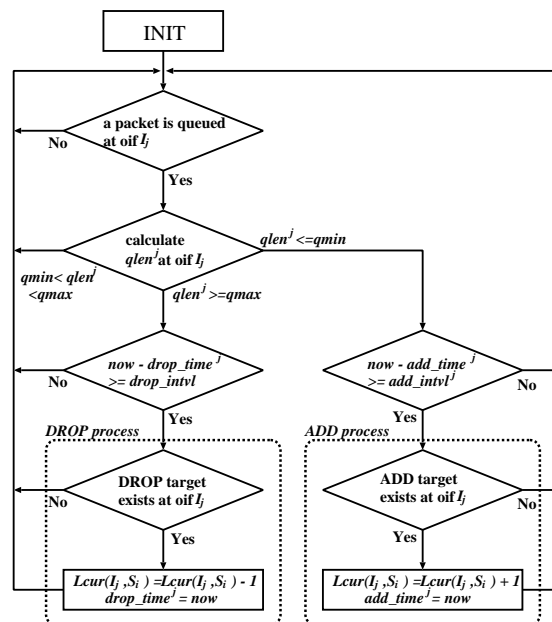


Figure 4. Filtering Algorithm

If $\max_k \{L_{cur}(I_j, S_k)\}$ equals to 1, there is no target layer because the base layer cannot be dropped. We express the number of the target layer of DROP as L_{dt} ($= L_{cur}(I_j, S_{i_0})$). When Layer(S_{i_0}, L_{dt}) is selected as the DROP target, $L_{cur}(I_j, S_{i_0})$ is decremented in the session state table. Thus the LMR begins to filter Layer(S_{i_0}, L_{dt}) at I_j . At the same time, the current time (now) is recorded in $drop_time^j$. After DROP is conducted, the LMR does not conduct DROP at I_j for $drop_intvl$ in order to wait the effect of the DROP to be reflected to $qlen^j$.

Next, we show ADD process. When the LMR detects the congestion solved at I_j , it searches the target layer of ADD from a session state table. The LMR selects the target layer of ADD which meets the following two conditions:

1. $i_1 = i$, if $L_{cur}(I_j, S_i) = \min_k \{L_{cur}(I_j, S_k)\}$.
2. $L_{cur}(I_j, S_{i_1}) + 1 \leq L_{max}(S_{i_1})$.

If S_{i_1} satisfying the first condition does not meet the second one, the LMR repeatedly searches another S_i except for S_{i_1} until it finds the target layer. We express the number of the target layer of ADD as L_{at} ($= L_{cur}(I_j, S_{i_1}) + 1$). When Layer(S_{i_1}, L_{at}) is selected as the ADD target, $L_{cur}(I_j, S_{i_1})$ is incremented in the session state table. Thus the LMR stops filtering of Layer(S_{i_1}, L_{at}) at I_j . At the same time, the current time is recorded in add_time^j . After ADD is conducted, the LMR does not conduct ADD at I_j for add_intvl^j . A LMR changes add_intvl^j adaptively to the network conditions in order to minimize the frequency of rate oscillation. Adaptive control algorithm for add_intvl^j

```

if ( $L \leq L_{cur}(I_j, S_i)$ ) {
   $L_{cur}(I_j, S_i) = L - 1$ ;
   $drop\_time^j = now$  ;
}
if ( $L > \max_{k \neq j} \{L_{cur}(I_k, S_i)\}$ )
  forward  $DROP\_REQ(S_i, L)$ ;
else
  discard  $DROP\_REQ(S_i, L)$ ;

```

Figure 5. Processing for $DROP_REQ(S_i, L)$

```

if ( $L > L_{cur}(I_j, S_i)$ ) {
   $L_{cur}(I_j, S_i) = L$ ;
   $add\_time^j = now$ ;
}
if ( $L > \max_{k \neq j} \{L_{cur}(I_k, S_i)\}$ )
  forward  $ADD\_REQ(S_i, L)$ ;
else
  discard  $ADD\_REQ(S_i, L)$ ;

```

Figure 6. Processing for $ADD_REQ(S_i, L)$

is described later in 3.6.

3.4. Signaling Protocol

The purpose of signaling is to prevent a LMR from transmitting a needless layer that results in unfair bandwidth allocation. $DROP_REQ(S_i, L)$ is used to require an upstream LMR to start filtering Layer(S_i, L) at the oif where the $DROP_REQ(S_i, L)$ is received. On the other hand, $ADD_REQ(S_i, L)$ is used to require an upstream LMR to transmit Layer(S_i, L) without filtering from the oif. These signaling packets are sent to a directly upstream LMR by unicast.

At first, we describe the generation of a signaling packet. When a LMR conducts DROP aimed at Layer(S_i, L_{dt}) or ADD aimed at Layer(S_i, L_{at}) at I_j , it generates $DROP_REQ(S_i, L_{dt})$ or $ADD_REQ(S_i, L_{at})$ if the following condition is met respectively:

$$L_{dt} > \max_{k \neq j} \{L_{cur}(I_k, S_i)\}.$$

$$L_{at} > \max_{k \neq j} \{L_{cur}(I_k, S_i)\}.$$

When it is necessary for the LMR to perform signaling, signaling packets are periodically generated at intervals of ss_intvl for $detect_period$ in order to assure the upstream LMRs to receive these packets even if some of them are dropped due to the congestion.

In order to cope with the congestion between LMRs that cannot be detected by LMRs, receivers detecting the congestion alternatively generate $DROP_REQ$ in the same way. Receivers use overall packet loss rate among all the layers

```

if ( $S_i$  is not registered in  $I_j$  record )
  register  $S_i$  in  $I_j$  record;
if ( $PrevID$  in  $S_i$  record  $\neq UpID$  in  $SESS$ )
   $PrevID = UpID$ ;
   $UpID =$  This LMR Address;
   $L_{max}(S_i) = L$ ;
if ( $now - add\_time^j \geq detect\_period$ ) {
  if ( $L > L_{cur}(I_j, S_i)$ )
     $L = L_{cur}(I_j, S_i)$ ;
  elseif ( $L < L_{cur}(I_j, S_i)$ )
     $L_{cur}(I_j, S_i) = L$ ;
}
forward  $SESS(S_i, L)$ ;

```

Figure 7. Processing for $SESS(S_i, L)$

of a session they join, and detect congestion when measured loss rate exceeds the threshold $loss_th$. On the other hand, a sender processes signaling packets in the same way as a LMR.

Next, we describe the processing for signaling packets. We show the processing for $DROP_REQ(S_i, L)$ and $ADD_REQ(S_i, L)$ in Figure 5 and Figure 6 respectively.

First, we show the processing for $DROP_REQ(S_i, L)$ when a LMR receives it at I_j . If $L \leq L_{cur}(I_j, S_i)$, this $DROP_REQ(S_i, L)$ is the first one received by the LMR and $L_{cur}(I_j, S_i)$ is decreased to $L - 1$. Then if $L > \max_{k \neq j} \{L_{cur}(I_k, S_i)\}$, the LMR is transmitting Layer(S_i, L) only from I_j and the layer is already unnecessary for all downstream receivers. Therefore the LMR forwards $DROP_REQ(S_i, L)$ upstream. Otherwise, the LMR discards $DROP_REQ(S_i, L)$.

Next, we show the processing for $ADD_REQ(S_i, L)$ when a LMR receives it at I_j . If $L > L_{cur}(I_j, S_i)$, this $ADD_REQ(S_i, L)$ is the first one received by the LMR, and the $L_{cur}(I_j, S_i)$ is increased to L . If $L > \max_{k \neq j} \{L_{cur}(I_k, S_i)\}$, the LMR is not receiving Layer(S_i, L) from the upstream LMR in spite of necessity to transmit it downstream. Therefore the LMR forwards $ADD_REQ(S_i, L)$ upstream. Otherwise, the LMR discards $ADD_REQ(S_i, L)$.

3.5. Session Control

The purpose of session control is to keep consistency of L_{max} and L_{cur} among session state tables of LMRs and to keep correct $PrevID$. Consistency of session state tables enables a LMR to select the proper target layer of DROP or ADD. A sender of S_i periodically multicasts $SESS(S_i, L)$ at intervals of ss_intvl when the sender is transmitting $L (= L_{cur}(I_j, S_i))$ layers.

Figure 7 illustrates the processing for $SESS(S_i, L)$ when a LMR transmits it from I_j . At first, the LMR registers S_i to the entry of I_j in the session state table if S_i has not been registered yet. Next, the LMR updates $PrevID$ to $UpID$ if their values are different, and then updates $UpID$ to the IP address of itself. This process enables a LMR to adapt to reconstruction of a multicast tree.

Then the LMR updates $L_{max}(I_j, S_i)$ to L because it is necessary for the LMR to know the available bandwidth at upstream links in order to avoid conducting unsuccessful ADD.

After that, if $now - add_time^j \geq detect_period$, the LMR updates L in $SESS$ or $L_{cur}(I_j, S_i)$. Until $detect_period$ passes after the last ADD, a LMR waits for $SESS(S_i, L)$ to reflect L_{cur} of upstream LMRs. If $L > L_{cur}(I_j, S_i)$, the LMR updates L to $L_{cur}(I_j, S_i)$. If $L < L_{cur}(I_j, S_i)$, the LMR updates $L_{cur}(I_j, S_i)$ to L . In this way, downstream LMRs are informed of the number of layers currently transmitted from upstream LMRs.

3.6. Extended Mechanism

In NLM, a LMR changes add_intvl^j adaptively to the network conditions in order to minimize the frequency of rate oscillation. When a LMR conducts ADD aimed at a layer for which enough bandwidth is not left on the path from the sender to receivers of the layer, consequently this ADD causes congestion that leads the LMR or another one to conduct DROP. Unless add_intvl^j is changed adaptively, this rate oscillation occurs at regular intervals.

A LMR changes add_intvl^j according to the following algorithm. If a LMR detects that congestion occurs somewhere in the network due to ADD conducted by the LMR at I_j , the LMR increases add_intvl^j as follows:

$$add_intvl^j = \min(add_intvl^j \times \alpha, add_intvl_max)$$

where $\alpha (> 1)$ is the increase constant. Otherwise, the LMR decreases add_intvl^j as follows:

$$add_intvl^j = \max(add_intvl^j \times \beta, add_intvl_min)$$

where $\beta (< 1)$ is the decrease constant.

A LMR recognizes that ADD causes congestion when one of the following conditions is satisfied. A LMR detects congestion at a downstream link or an upstream one by the second condition or the third one respectively.

- The LMR detect congestion at I_j within $detect_period$.
- The LMR receives $DROP_REQ(S_i, L_{cur}(I_j, S_i))$ at I_j within $detect_period$.
- The LMR does not receive $SESS(S_i, L_{cur}(I_j, S_i))$ after $detect_period$ has passed since the last ADD.

Table 2. Simulation Parameters

$qmax$	15	[packets]
$qmin$	3	[packets]
$qweight$	0.05	
add_intvl_min	5	[s]
add_intvl_max	80	[s]
$drop_intvl$	0.5	[s]
$detect_period$	5	[s]
α	2.0	
β	0.75	
$loss_th$	0.25	
ss_intvl	0.1	[s]

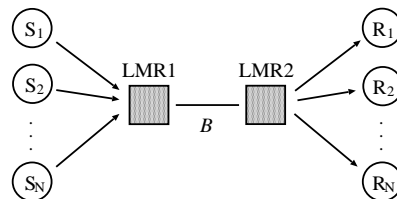


Figure 8. Simulation Topology

4. Simulations

In this section, we present simulation results to explore the performance of NLM. We implemented NLM in the LBNL network simulator *ns*[16]. We evaluated the performance of NLM comparing with RLM[4] that is the most general receiver-driven rate control mechanism for layered multicast.

4.1. Simulation Model

All the parameters used in our simulations are summarized in Table 2. Figure 8 illustrates the simulation topology where N single-source single-receiver sessions share the bottleneck link. Bandwidth of the bottleneck link is B [Mbps], and propagation delay of each link is 10 [ms]. Output queue size of each LMR is 20 [Packets]. The maximum number of transmitted layers is 5. The rate of each layer is CBR and 100, 100, 200, 400, 800 [Kbps] from the base layer respectively. Therefore the maximum transmission rate is 1.6 [Mbps]. The size of a data packet is 1024 [Byte]. We ran each simulation for 10 minutes.

In our simulation, we took into account only the packets dropped by drop-tail in spite of passing the filter without being filtered to calculate packet loss rate of NLM. We did

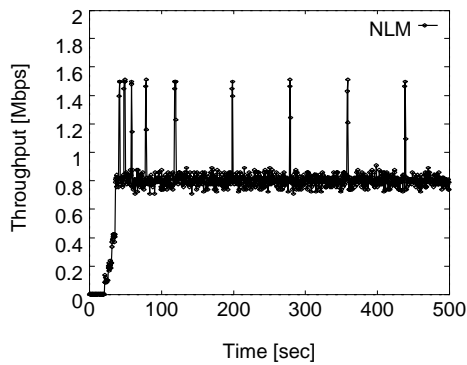


Figure 9. Behavior of Rate Control

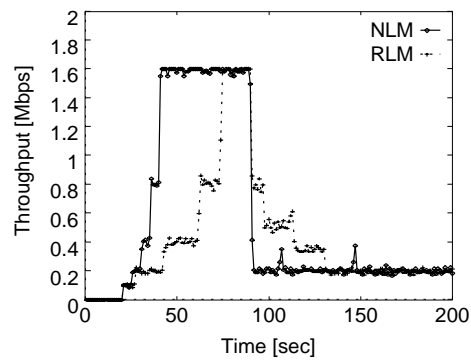


Figure 11. Response to the Congestion

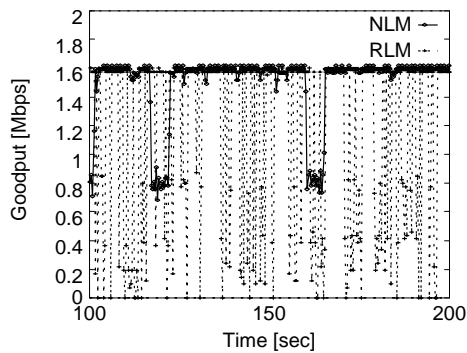


Figure 10. Goodput

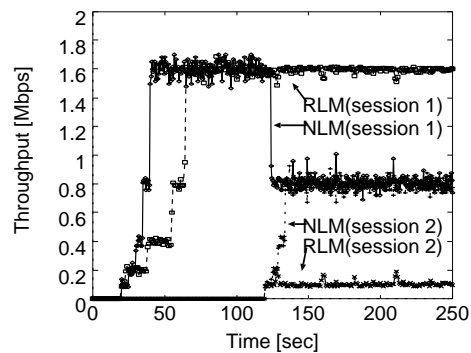


Figure 12. Fair Bandwidth Allocation

not consider the leave delay caused by IGMP. Note that this condition is preferable for receiver-driven rate control.

4.2. Simulation Results

Figure 9 presents the behavior of rate control. In this simulation, variables have the following values : $N = 1$, $B = 1.5$ [Mbps]. A receiver joins session S_1 at 20[s] and a sender increases transmission rate until LMR1 detects congestion and sends $DROP_REQ(S_1, 5)$ to the sender at 41[s]. Since B is not enough for 5 layers, LMR1 increases add_intvl^j gradually up to add_intvl_max to decrease the frequency of rate oscillation. In addition, NLM keeps packet loss rate 0[%] through this simulation because LMR1 can react to the congestion quickly before packets are dropped by drop-tail.

Figure 10 shows goodput, which reflects apparently the received quality in layered multicast. We define goodput as the sum of throughput of the layers which can be received with less than 20[%] packet loss rate in order from the base layer. In this simulation, variables have the following values : $N = 1$, $B = 1.6$ [Mbps]. Through this simulation,

the maximum overall packet loss rate is 9[%]. Since RLM detects congestion by measured packet loss rate and set the threshold to 25[%], RLM does not detect congestion and keeps the subscription level 5. However, obtained goodput of RLM is not stable and is 0 [Mbps] in the worst case when the packets on a base layer are dropped. On the other hand, since NLM detects congestion by measured queue length, LMR1 sometimes conducts DROP in advance to avoid packet losses. Thus NLM can achieve stable and as high goodput as throughput.

Figure 11 shows the response to the congestion. In this simulation, variables have the following values : $N = 1$, $B = 1.6$ [Mbps]. A receiver joins session S_1 at 20[s] and non-adaptive CBR cross traffic with 1.3[Mbps] is generated at 90[s]. When cross traffic is generated, remaining bandwidth is only 0.3[Mbps] and the optimal subscription level is 2. RLM detects congestion at 98[s], 114[s], and 130[s], and drops Layer($S_1, 5$), Layer($S_1, 4$), and Layer($S_1, 3$) respectively. Thus RLM takes 40[s] to solve the congestion because of the estimation delay. Due to this slow response to the congestion, suffered packet loss rate of RLM is as

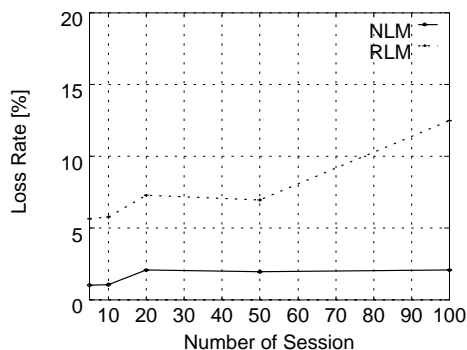


Figure 13. Session Scalability

high as 50[%] just after cross traffic is generated and RLM keeps high loss rate (more than 10[%]) for 40[s]. On the other hand, NLM quickly responds to the congestion within 2[s], and suffered packet loss rate is only 1[%]. In this way, NLM can keep low packet loss rate even in such a service model that enormous cross traffic is generated suddenly.

Figure 12 shows the ability of NLM to achieve fairness among competing sessions which start at different time. In this simulation, variables have the following values : $N = 2$, $B = 1.7$ [Mbps]. The receiver of session S_1 and that of session S_2 join at 20[s] and 120[s] respectively. In RLM, S_2 cannot get the fair share (0.8 [Mbps] in this simulation) because the receiver of the ongoing and stable session (S_1) does not drop Layer(S_1 , 5) due to estimation delay. On the other hand, NLM can achieve fairness because LMR1 filters Layer(S_1 , 5) in session S_1 to which more bandwidth is allocated than S_2 .

Figure 13 shows session scalability of NLM. In this simulation, we varied N from 1 to 100 and set B as follows : $B = 1.0 \times N$ [Mbps]. We plotted the maximum packet loss rate. In RLM, suffered packet loss rate increases as the number of sessions increases. On the other hand, NLM can keep it only 2[%] regardless of the number of sessions because NLM can react to the congestion quickly.

5. Conclusion

In this paper, we presented a network-supported rate control scheme for layered multicast (NLM). NLM can avoid packet losses on a lower layer and achieve fairness. In NLM, a router with rate control functions (LMR) filters a layer with the lowest priority among sessions when it detects congestion or filters a needless layer when the LMR is informed by signaling that the layer is filtered further downstream. Simulation results presented in this paper indicate good responsiveness, fairness and stability of goodput of NLM compared with a receiver-driven scheme.

References

- [1] T. Turetti and J.-C. Bolot. Issues with multicast video distribution in heterogeneous networks. In *Proc. of 6th International Workshop on Packet Video*, Oct. 1994.
- [2] S.Y. Cheung, M. Ammar, and X. Li. On the Use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution". In *Proc. of IEEE INFOCOM'96*, Mar. 1996.
- [3] E. Amir, S. McCanne, and H. Zhang. An Application-level Video Gateway. In *proc. of ACM Multimedia'95*, Nov. 1995.
- [4] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven Layered Multicast. In *Proc. of ACM SIGCOMM'96*. Aug. 1996.
- [5] X. Li, S. Paul, and M. Ammar. Layered Video Multicast with Retransmissions (LVMR): Evaluation of Hierarchical Rate Control. In *Proc. of IEEE INFOCOM'98*, Mar. 1998.
- [6] X. Li, S. Paul, and M. H. Ammar. Multi-session Rate Control for Layered Video Multicast. In *Proc. of Multimedia Computing and Networking 1999*, Jan. 1999.
- [7] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like congestion control for layered multicast data transfer. In *Proc. of IEEE INFOCOM'98*, Mar. 1998.
- [8] L. Wu, R. Sharma, and B. Smith. Thin Streams: An Architecture for Multicasting Layered Video. In *Proc. of NOSS-DAV'97*, May 1997.
- [9] T. Turetti, S.F. Parisi, and J.-C. Bolot. Experiments with a Layered Transmission Scheme over the Internet. INRIA Research Report No. 3296, Nov. 1997.
- [10] S. Bhattacharjee, K. L. Calvert, and E. W. Zegura. Network Support for Multicast Video Distribution. Technical Report (GIT-CC-98-16), 1998.
- [11] D. Rubenstein, J. Kurose, and D. Towsley. The Impact of Multicast Layering on Network Fairness. In *Proc. of ACM SIGCOMM'99*, Sept. 1999.
- [12] S. Floyd and V. Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, Vol.1, No.4, pp.397-413, Aug. 1993.
- [13] M. Grossglauser and J.-C. Bolot. On Service Models for Multicast Transmission in Heterogeneous Environments. In *Proc. of IEEE INFOCOM 2000*, Mar. 2000.
- [14] R. Gopalakrishnan, et al. Stability and Fairness Issues in Layered Multicast. In *Proc. of NOSS-DAV'99*, June 1999.
- [15] S. Bajaj, L. Breslau, and S. Shenker. Uniform versus Priority Dropping for Layered Video. In *Proc. of ACM SIGCOMM'98*, Sept. 1998.
- [16] UCB/LBNL/VINT Network Simulator – ns (version 2). <http://www-mash.cs.berkeley.edu/ns/>, Sept. 1997.