IJCSN International Journal of Computer Science and Network, Volume 3, Issue 1, February 2014
ISSN (Online) : 2277-5420
www.IJCSN.org

A Congestion Control Scheme for Layered Video Streaming over the Internet

[1]Hamid Reza Barzeghar, [2]Dr S. Khursandi

[1]Department of Electrical, Computer and IT Engineering,
Qazvin Branch, Islamic Azad University, Qazvin, Iran

Abstract - The Multi-rate Multicast Congestion Control(MR-MCC) scheme has been considered as a suitable scheme for video multicasting for a very large heterogeneous group of receivers. This paper presents an approach to Multi-rate Multicast Congestion Control for real-time, loss-sensitive, video layered multicast applications. The basic idea of the algorithm is to react to incipient congestion before packet loss occurs by monitoring variations in the one-way packet delay from sender to receivers. we present a novel congestion control technique using a hybrid rate plus window based rate control to minimize queuing delay and packet loss. In our methodology, we do not try to explicitly measure the available network bandwidth but rather use a congestion level measure. Furthermore, by constraining the bandwidth of the layers to a well-defined rate, the congestion control can be accomplished almost entirely without packet loss. This is particularly suitable for real-time multimedia conferencing applications that are inherently multipoint and loss-sensitive. The performance of the Congestion control algorithm in terms of link utilization, inter- and intra-protocol fairness, session scalability and TCP-friendliness is evaluated through extensive simulation.

Keywords - *Rate adaptation, Mufti-rate Multicast, Packet-pair probe, TCP-friendliness, TCP throughput equation*

1. Introduction

There are two directions of multicast congestion control schemes proposed so far, namely Single-rate Multicast Congestion Control (SR-MCC) and Multi-rate, Multicast Congestion Control (MR-MCC). To achieve the scalability for a very large heterogeneous group of receivers over the Internet, MR-MCC is a better choice as giving more flexibility in allocating of bandwidth along different network paths.

Recently, Several protocol studies (such as [1], [2], [3], [4]) have focused on the design of MR-MCC protocols.

However, all of them have some drawbacks. Some designs cause over-subscription and high packet losses. Some are slow convergent and unresponsive. Some are TCP-unfriendly. Some designs are too complex and infeasible.

Hence, in this paper, we propose a new design of MR-MCC, which has the following properties: scalability, responsiveness, fast- convergence, TCP-friendliness, efficiency in network utilization and simplicity to implement.

The basic idea of the algorithm is to react to incipient congestion before packet loss occurs by monitoring variations in the one-way packet delay from sender to receivers and using TCP throughput equation. We present a novel congestion control technique using a hybrid rate plus window based rate control to minimize queuing delay and packet loss. It uses AIMD to guarantee fairness across multiple flows but uses fast ramp-up and graceful back-off to prevent link under utilization caused by lower congestion detection thresholds. The use of pacing and lower congestion detection thresholds allows us to control queuing delay to desired levels and is inspired by the work in [12].

In order for the to congestion control algorithm to be able to respond to congestion before packet loss occurs, the variations in packet transmission delay can be used to detect congestion. An increasing delay indicates that router buffers are filling up and must be responded to by lowering the effective bandwidth. Similarly a delay that has decreased below some threshold indicates that it might be possible to increase the bandwidth. To avoid packet loss the increase in bandwidth resulting from joining an additional group must be small enough for the network to buffer the excessive packets for the time it takes the receivers to detect the congestion and respond to it by leaving the group and is inspired by the work in [13].

In our way, each affected receiver decrease its reception rate by dropping a layer in multi–layered multicast. On the other hand, any receiver can increase the reception rate (by joining an additional enhancement layer) when it experiments favorable network conditions (e.g., no loss detected during a certain period). The remainder of this paper is organized as follows. We begin in Section II discussing some related background. In Section III, we describe our design goals and principle. Section IV shows the performance evaluation of our protocol using a network simulator (ns2). Finally, in Section VII, the conclusions of this work are given.

2. Background and Related Work

This Section provides the background of the related issues to be covered in this paper.

2.1 Receiver–Driven Layered Multicast (RLM)

The RLM protocol is a simple control loop where the source does not take any active role while the protocol machinery is run at each receiver [5]. The source encodes its signal into layers and transmits each layer on a distinct multicast group. Each receiver participating in the session drops a layer (i.e., leaves the associated group) on congestion or adds a layer(i.e., joins the associated group) when scarce bandwidth is available non-necessarily in a cumulative way.

While congestion is expressed in packet being lost and poor quality, the absence of an equivalent signal when the level of subscription is too low imposes to carry out active capacity inference through join–experiment. To do this, a receiver spontaneously joins the next layer; if congestion occurs, the layer just added is dropped. As failed join–experiments cause transient congestion, an exponential backoff is applied to problematic layers. A shared learning algorithm is also used to allow proximate receivers to learn from each other by listening to on going join–experiments in the group. This avoids backing off of one's join–timer in reaction to a congestion unrelated to one's join–experiment at the cost a notification message to be sent to the group.

2.2 Receiver–driven Layered Control (RLC)

Vicisano et. al. proposed in [6] a TCP-like congestion control for layered multicast suitable for continuous stream and reliable bulk data transfer in the Mbone primarily. As in RLM ,the source encodes data in layers and transmits it to receivers which modulate their respective reception rate by joining or leaving these layers. The join and leave strategy is chosen to emulate the behavior of TCP and achieve a throughput for a given layer proportional to the inverse of the square–root of loss rate.

To prevent inefficient uncoordinated actions of receivers behind a common bottleneck, an implicit coordination is done using synchronization points (SP). A SP corresponds to a special flagged packet in the data stream and SPs at each layer are always a subset of the SPs at the previous layer. This permits receivers with low subscription level (i.e., having low reception rate) to increase more quickly their reception quality as SPs are scarce in higher layers. A receiver can only attempt a join immediately after an SP. The sender periodically (and prior to SP) generates short bursts of packets to probe for bandwidth; a long relaxation period follows where no packet is sent. If the probe packets does not lead to congestion, there is confidence that subsequent join attempt can be successful. The protocol thus consists in a receiver decreasing its subscription layer if a loss is experienced during normal transmission, increasing it at a SP if no loss occurs during the preceding burst period.

2.3 Packet–Pair Layered Multicast (PLM)

Legout and Biersack outlined a pathological behavior of both RLM and RLC in [3] resulting in instability and periodic losses and due to bandwidth inference through join–experiments or prob bursts. To avoid this inherent - and difficult to correct – pathologicaln behavior, they proposed in [3] the PLM protocol which uses Packet-Pair instead to infer the available bandwidth. PLM requires all routers to provide a fair queuing functionality.

2.4 Layered Video Multicast with Retransmission (LVMR)

LVMR [7] uses a layered MPEG-encoded transmission and a hierarchy protocol of agents distributed in the network to perform the congestion control. The agents coordinate join and leave decisions of receivers based on static loss threshold (original version) or using a simple TCP- equation to achieve friendliness. Retransmission is done by designated local receivers to reduce recovery time. The simulations report improvement over RLM but the usage of control agents introduces considerable overhead even though it frees the sender from the burden of congestion control.
Agents are more easily implemented at application level compared to the cost of requiring some news features from internal network nodes. Thus, LVMR can be

successfully incorporated in an Application Layer Multicast (ALM) environment but one should keep in mind that the overall system performance rely heavily on the overlay network organization which is harden further in the multirate context.

2.5 Fine–Grained Layered Increase/Decrease with Dynamic Layering (FLID–DL)

FLID–DL [1] provides a generalization of RLC through a dynamic layering strategy to avoid IGMP leave latency bottleneck. The implementation uses a digital fountain encoding allowing a receiver to recover the original data upon reception of a fixed number of distinct packets, regardless of which layers it subscribed over time. To avoid long leave latency limiting the responsiveness to congestion, a layer transmission rate decreases over time; the layers bandwidth distribution is flexible (uniform, heavy-tail).

With such a dynamic layering, a receiver can reduce its reception rate simply by not joining any additional layer. On the other hand, to maintain or increase its reception rate, the receiver should join more layers. Receivers join and leave actions to adjust their reception rates are based on congestion conditions through TCP–like throughput equation with a fixed RTT (hence potentially unfair).

The sender places signals into packets to dictate join and leave behavior of receivers in order to avoid inefficiency due to uncoordinated actions of hosts sharing a bottleneck link.

To do this, time is partitioned into time slots by the sender and a receiver may increase its reception rate at the beginning of the next time slot if no packet loss occurs during the current time slot.

3. Framework and Algorithms

In this section, we explain the fundamental concepts of our design, assumptions, and requirements of the protocols.

3.1 Layered Data Organization

In a layered multicast session a source emits data to a set of cumulative multicast groups called layers. Each layer carries data with bandwidth Li in such a way that the cumulative data rate on layer i is:

$$B_i = \sum_{j=0}^{i} L_j \qquad (1)$$

We call this cumulative layered data organization. In cumulative layered data organization the sender must emit data such that the cumulative data carried on layer i is a subset of cumulative data carried on layer j for all i<j. That is, the base layer carries the minimum set of data to deliver. The second layer delivers additional data that may be decoded in conjunction with the base layer. The third layer provides additional data, and so on. All layers together offer the maximum set of data to deliver.

In layered multicast congestion control approach the congestion control is performed by receivers through subscription level management. In cumulative layered data organization the order in which layers are subscribed is predefined.

Sometimes layered multicast congestion control is called multi-rate congestion control. We use the term layered and interpret it as an instance of general multirate, where there is possibly no predefined order of subscription.

3.2 Subscription Level Management

In receiver-driven layered multicast congestion control the sender emits data to multicast groups called layers, in a cumulative manner. The cumulative order of layers is predefined and receivers perform congestion control by subscribing and unsubscribing layers according to the congestion in the network. A subset of layers subscribed by a receiver is referred to as subscription level. While a unicast flow can adapt the sending rate with high granularity, in a layered multicast approach the receiver may only choose from relatively coarse grained rates, i.e., from the layers. This leads to a dilemma of fair competition, oscillations and claim of fair share. When the fair rate is between layers, a recipient may only choose from a layer above the fair rate (too high) or from a layer that is less than the fair share (too low). Long-term fair share could be achieved by oscillating between the two, but oscillations are considered bad. Thus, in a receiver-driven layered congestion control approach, there are scenarios where all of the three are tremendously hard to reach. Performance of a layered multicast congestion control depends much on the management of the subscription level at receivers. We call this subscription level management. There is a possible strategy to perform this activity.

3.3 Management Strategy

The objective of this strategy is to stick to the latest decision when possible, and to avoid changes in the subscription level. To do so, Instead of directly subscribing the next layer, or leaving the current layer, we trigger join and leave timers. More precisely, when the calculated rate becomes greater than Bi+1, we trigger a join-timer that expires after Tjoin seconds. When the timer fires, layer Bi+1 is subscribed. Similarly, when calculated rate becomes less than Bi, we trigger a leave-timer that expires after leave second. When the timer fires, layer Bi is unsubscribed. If the calculated rate drops below Bi+1 before the timer join has fired, the join-timer is canceled. Similarly, if the calculated rate rises above the Bi before the leave-timer fires, the leave-timer is canceled[13].

However, we define a passive zone in which the timers are not triggered at all. The passive zone spans slightly over the current and next layer, i.e., slightly below the currently subscribed layer i and slightly beyond the next layer i+1. Figure 1 illustrates the passive zone.
When the calculated rate enters or exits the passive zone, a timer is triggered.That is, when the calculated rate falls below the passive zone (and thus below the currently subscribed layer i, too), a leave timer is triggered. Similarly, when the calculated rate gets above the passive zone (and thus above the next layer i+1), a join timer is triggered. However, the triggered timers are constantly updated according to calculated rate.

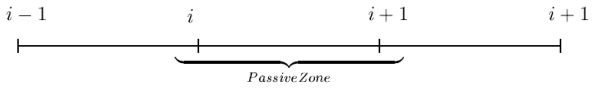


Figure 1: Passive Zone of strategy

The basic idea is to make decisions When the calculated rate is only slightly above the next layer, a waiting time close to the maximum waiting time is computed until the layer is joined. The same applies to leaving: when the calculated rate is only slightly below the current layer, a leave delay close to the maximum leave delay is computed. However, if the calculated rate gets far above the next layer, or far below the current layer, the timers are adjusted short.

We linearly map the calculated rate (Xcalc)distance to the layer rates Bi or Bi+1 to get delta values in the range [0,1]. That is, if Xcalc is exactly in the middle of Bi-1and Bi, then delta value for leaving is 0.5. Eq. 2 and 3 show how to compute the delta values for joining and leaving, respectively[13].

$$\delta_{join} = \frac{X_{calc} - B_{i+1}}{B_{i+1} - B_i} \qquad (2)$$: delta values for joining

$$\delta_{leave} = \frac{B_i - X_{calc}}{B_i - B_{i-1}} \qquad (3)$$: delta values for leaving

The join reactivity functions are *djoin=1-δjoin* for joining and *dleave=1-√δlevae* for leaving.Thus, the system is more reactive when it comes to leaving a layer compared to joining a layer. Depending on the application, different functions may be appropriate.
And:

Timer-join=(1-djoin)*(Tmax) (4)

Timer-leave=(1-√D)*(Tmax) (5)

Where Tmax is maximum Average one-way delay packets in through session.

3.4 Slow-Start

In environments with low degree of statistical multiplexing this poses a challenge. At startup, when layers are expected to be gradually subscribed, the calculated rate would prevent the receiver from subscribing further layers (in the beginning the data rate is low until additional layers are subscribed).

To avoid this problem we implemented a slow-start phase which attempts to bring the receiver to the equilibrium at startup. A receiver always starts in the slow-start and increases the subscription level until the equilibrium has been reached. The convergence speed can be adjusted with a parameter slow-start interval which we refer to as I. It is the time in seconds that must elapse from previous join before performing the second join. The delay applies to all layers, i.e., independent which layer is being joined, the delay is always constant. The interval is also used to adjust a deaf-period. Deaf-period means the time, after slow-start join, during which the calculated rate is not consulted. The deaf-period is proportional to the layer bandwidth[13]:

$$D_i = I \cdot \frac{B_0}{B_i} \qquad (6)$$

Where Di is the deaf-period after joining layer i, Bo is the base layer bandwidth and Bi is the cumulative bandwidth on layer i.

Once the deaf-period has elapsed, if the calculated rate is below Bi, then the slow-start terminates and normal operation is resumed. Thus, the deaf-period becomes shorter as the cumulative bandwidth increases. In our simulations this slow-start algorithms appears to give fairly good results.

However, a thorough analysis would be required to make more general conclusions of its performance. We expect it to be a good start for optimization and further study. Especially the deaf-period requires further study. Although it is directly adjusted by layer rate ratio, and, thus, not really dependent on the layer granularity, it might show bad performance when the layer granularity is set to linear instead of exponential.

3.5 Delay Trend Detection

In recent years, the one-way delay trend detection techniques have been proposed to estimate end-to-end available bandwidth. However, various error sources may distort the measured delay trend. Instead of trying to estimate the absolute available bandwidth, [8] provides a delay trend detection method to enable a receiver to judge whether the available bandwidth is at or higher than the target layered bit rate.

3.5.1 Delay Trend Model

Suppose there are N links from the sender to the receiver. The one-way delay of a packet with size L_k is:

$$D_k = \sum_{i=1}^{N} \left(\frac{L_k}{c_i} + d_i^k + \sigma_i\right) D_k = \sum_{i=1}^{N} \left(\frac{L_k}{c_i} + d_i^k + \sigma_i\right) \quad (7)$$

Where σ_i is the processing delay at router i ; d_i^k is the queuing delay at router i . D_k is a random variable depending on the packet size and the network condition (queuing delay). A delay increase trend exists when the following delay relationship holds:

$$p(D_l > D_m) > 0.5 \qquad \forall l > m \quad (8)$$

Considering that the packet size is variable, there are two factors can contribute to a delay increase trend. A packet size increase trend alone may cause the observed one-way delay to have an increase trend. The network congestion may also contribute to the delay increase trend. Eq. (8) can be rewritten to:

$$D_k = \left(\sum_{i=1}^{N} \left(\left(\frac{1}{c_i}\right) L_k\right) + \sum_{i=1}^{N} d_i^k + \sum_{i=1}^{N} \sigma_i\right) \quad (9)$$

where $\sum_{i=1}^{N} \frac{1}{C_i}$ is a constant if the path doesn't change and $\sum_{i=1}^{N} \sigma_i$ is the total router processing delay, which is a random variable independent to packet number k . When the packets arrive at router i below its capacity Ci , the queue at the router will not build up and a later coming packet will see the same number of packets in the queue as the previous packets by probability.

In this case, the queuing delay d_i^k can be considered independent to packet number k and doesn't contribute to a delay increase trend.

On the other hand, if the packets arrive at router i at or above its capacity Ci , the queue at the router will build up and a later coming packet will see more number of packets in the queue than the previous packets by probability. The queuing delay d_i^k now shows correlation with the packet numbers k , i.e., later packets tend to have longer queuing delays than previous packets. Finally if one of the routers along the path is congested, the second term in Eq. (9), which is the summary of queuing delay at all routers, will contribute to the one-way delay increase trend.

So we can infer whether the packet sending rate exceed av-bw by detecting delay increase trend and packet size trend. We use the following logic to do the estimation:

$$\begin{cases} rate < av-bw, & S\_trend \geq 0 \,\&\, D\_trend < 0 \\ rate > av-bw, & S\_trend \leq 0 \,\&\, D\_trend > 0 \\ rate \approx av-bw, & S\_trend = 0 \,\&\, D\_trend = 0 \\ don't\ know, & others \end{cases}$$

where S _ trend represents packet size trend while D_ trend represents one-way delay trend. Trend > 0 shows an increase trend, trend < 0 shows a decrease trend and trend = 0 shows no trend.

3.5.2 Trend Detection

We use bi-weighted moving protocol to detect trend one-way delay. After we get the one-way delay D_k (packet size

trend is processed in a similar way), we apply moving average (MD_k) to them:

$$TD_{k=MD_k} * (1-\alpha) + D_{k+\alpha} \qquad k>0$$
$$MD_k = \frac{1}{k}\sum_{i=1}^{k} D_i$$

$$OWDi=(TD_k - MinTD_k) \qquad i=2,\ldots,m \qquad (10)$$

Where α is a parameter controlling the weight of latest delay in MD_k.In this paper, we use α =0.1.The benefit of using bi-weighted moving average is that all measurement data are used and the computation overhead is stillo (N) . $MinTD_k$ and OWDi are minimum average one-way delay and variations in the one-way packet delay of these packets. Our approach is better than the Pairwise Comparison Test (PCT) and Pairwise Difference Test (PDT) of the delay trend detection in Pathload[9].

Our algorithm is more robust. For example, if the available bandwidth suddenly drops down during the estimation period, the Pathload or IGI may overestimate the available bandwidth and thus, it is not appropriate for multimedia streaming. However, our probing rate will not be influenced by the sudden decrease of available bandwidth. On the other way, when available bandwidth increases during the probing, the estimation might be underestimated but it will not induce packets lost.

We use four metrics is called the TE1,TE2,TE3,TE1,1, which examines the rising strength of then OWDs trend.

$$TE1 = \frac{\sum_{m=2}^{n}\sum_{i=1}^{m-1} I(OWDm > OWDi)}{\frac{n(n-2)}{2}}$$

$$TE2 = \frac{\sum_{i=2}^{n} I(OWDi+1 > OWDi \text{ and } OWDi > OWDi-1)}{(n-2)}$$

$$TE1,1 = \frac{\sum_{i=2}^{n} I(OWDi > OWDi-1)}{N-1}$$

$$TE3 = \frac{\sum_{i=2}^{n} I(OWDi+1 < OWDi \text{ and } OWDi < OWDi-1)}{n-2}$$

where OWDi is variations in the one-way delay i and I(X) is 1 if X holds and 0 otherwise.
Other metrics are:

$$TD1 = \frac{TE1 + TE1,1}{2}$$
$$TD2 = \frac{TE2 + TE3}{2} \qquad (11)$$

We modify the full search algorithm to detect the delay trend as:

IF 0.4≤TD2≤ 0.5 THEN
IF TD1≥0.5 THEN
Increasing trend.
ELSE IF 0.4≤TD1 THEN
Decreasing trend.
ELSE IF 0.4<TD1<0.5
No trend.
End if
Else
Delay trend unnormal.
End if

We use a trend to determine the trend of the packet size :

$$TS = \frac{\sum_{i=2}^{n} I(dsowi > dsowi-1)}{n-1} \qquad (12)$$

where *dsow* is the the packet size and I(X) is 1 if X holds and 0 otherwise.

Search algorithm to detect the packet size trend as:

IF Ts≥0.5 THEN
Increasing trend.
ELSE IF 0.4≤TS THEN
Decreasing trend.
ELSE IF 0.4<TS<0.5
No trend.
End if

The trend threshold (that can be used to check whether there is an increasing trend) of TD1 is 0.5 and 0.4.If the test result is larger than threshold, the measurements have an increasing trend, otherwise no increasing trend.

3.6 Rate Update

We then classify the congestion level into one of the following three zones.

- Zone 1: OWD trend is non-increasing and dsow trend is non-increasing and average queuing delay is less than some threshold (daowd $\leq$d1).
- Zone 2: OWD trend is non-increasing, no packet is lost and dsow trend is non-increasing, and d1 < daowd $\leq$d2, for d2 > d1.
- Zone 3: OWD trend is increasing and dsow trend is increasing, daowd > d2, or packet

If packets are being properly paced, an increasing OWD and dsow trend means buffers are building up and thus implies congestion. The use of delay in determining congestion level is a commonly used technique. However, our classification of congestion level into three zones is unique, as is the use of OWD and dsow trend as an additional indicator of congestion.

The zone classification has an intuitive explanation. Ideally,we would always like to stay in Zone 2 (the zone which give the tolerable queuing delay) and thus the typical queuing delay seen is between d1 and d2. Going to Zone 1 results in a queuing delay below what we want and lower link utilization[12].

Zone 3 results in higher queuing delay and probability of congestion induced loss. Thus delays larger than d2 will only be seen when new flows enter. By appropriately choosing d1 and d2 and accounting for typical propagation delay seen on the link, the end-to-end delay due to the network can be controlled. Note that we do not back off due to packet loss unless it is accompanied by a delay increase. This allows us to not be as sensitive to random losses, such as caused by wireless links.At the end of every epoch, the transmission rate (R) is updated based on the congestion classification.

Instead of updating the window, the transmission rate is directly updated using:

$$Rtarget = \begin{cases} Rcurr + \alpha & \text{if zone=zone1 or zone 2} \\ Rcurr(1-\beta) & \text{if zone=zone >3} \end{cases}$$

$$\alpha = \begin{cases} \alpha\max & \text{if daowd<d0} \\ & \text{else} \\ \dfrac{\alpha\min\ \alpha\max(d1-d0)}{\alpha\max(daowd-d0)+\alpha\min(d1-daowd)} & \end{cases}$$

$$\beta = \begin{cases} \beta\min+\dfrac{\beta mid-\beta\min}{d2-d1}(daowd-d1) & \text{if zone=zone3} \\ \beta\min+\dfrac{\beta\max-\beta mid}{d3-d2}(daowd-d2) & \text{if zone=zone3 \& daowd incr} \\ & \text{\& daowd=<d3} \\ \beta\min+\dfrac{\beta\max-\beta mid}{d3}(daowd) & \text{if daowd incr} \\ & \text{\& daowd=<d3} \\ \beta\max & \text{if daowd>d3} \end{cases}$$

Figure 2: Passive Zone of strategy

βmin, βmid, and βmax are used to control the shape of the β curve. β goes from βmin to βmid during Zone 2, and then up to βmax in Zone 3 if the delay trend is non-increasing. If the delay trend is increasing, then it is assumed to be a sign of congestion and β linearly increases as a function of delay up to βmax regardless of queuing delay. For cases where packet loss is encountered and daowd > d1, β = βmax.

The receiver estimates the target reception rate (RTARGET)as follows:

If (RTARGET ≥ 0) Then
Set RTARGET = Min(RTCP ,RTARGET)
Else If (RTARGET = -1) Then
Set RTARGET = RTCP
End If
Else
Set RTARGET = Rtarget
End if

The receiver subscribes to and unsubscribes from layers according to the RTARGET follows:
If (Ri> RTARGET) Then
Repeat Until (Ri≤ RTARGET)
If i > 0 Then
If Timer-leave ==0 Then

Unsubscribe from a layer

i = i -1
Timer-leave = l
Else
Exit loop
Else
EXIT the session
End If
Loop
Else If (Ri< RTARGET)
Do While (RTARGET <Ri)
If Timer-join=0 Then

Subscribe to a layer
i = i +1
Timer-join = (n/N+random(0,0.5))l
Else
Exit loop
Loop
Else If (Ri = RTARGET)
Maintain the current subscription level
End If

Where RTCP, RTARGET are TCP friendly Rate, calculates an estimated rate in receiver.

H.TCP-Friendly Rate Estimation

In our protocol usually implement a simple AIMD (Additive Increase, MultiplicativeDecrease) as TCP. The sending rate may be adjusted to the TCP–friendly throughputin the same network conditions based on equations modeling TCP behavior for fairnesspurpose. Padhye et. a1 [10] have proposed a better model for a broader range of network conditions. The model is given as:

$$R_{TCP} = \frac{8M}{t_{RTT}\sqrt{\frac{2bl}{3}} + t_{RTO}\,Min\,(1,3\sqrt{\frac{3bl}{8}})l(1+32l^2)}$$

where b is the number of packets acknowledged by each ACK, RTT is the TCP retransmission timeout (in seconds).This model is for TCP Reno. It is the most widely accepted TCP throughput model by the Internet research community. To calculate the TCP-friendly rate in our algorithms, we simplify as[11] :

$$R_{TCP} = \frac{8M}{T_{RTT}\sqrt{\frac{2l}{3}} + 12\sqrt{\frac{3l}{8}}l(1+32l^2)}$$

where M is the packet size (in bytes), T rtt is the RTT (in seconds), and l is thePLR (between 0.0 and 1.0).

4. Simulation Experiment

In this section, we discuss simulation scenarios, objectives, and results of experiment sets used to evaluate our style. We start from Experiment I testing how fast and with what stability our protocol converges to an optimal rate. Experiment II tests intra-protocol fairness. For the protocol, we use αmin = 800bps, αmax = 40Kbps, βmin = 0.25, βmid = 0.33, βmax = 0.5 and:

dm=daowd max-daowdmin
di=ηidm , ηi=(0.1,0.25,0.8)

These parameters are found by tuning for our queuing delay requirements.

Experiment I: Convergence to Target Rate

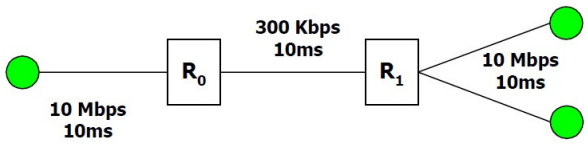


Figure 3: Simulation Topology of our protocol Experiment I.

The objective of this experiment is to test how our protocol converges to the target rate, and to verify that our protocol receivers can estimate the available bandwidth properly and adjust the subscription level to the optimal level quickly. We use the topology depicted in Fig. 3 of a single multicast source with two receivers. The input bandwidth is 10 Mbps, while the bottleneck link is 300 Kbps. We start the multicast source at time zero and its sinks randomly three seconds later. The simulation is run for 80 seconds.

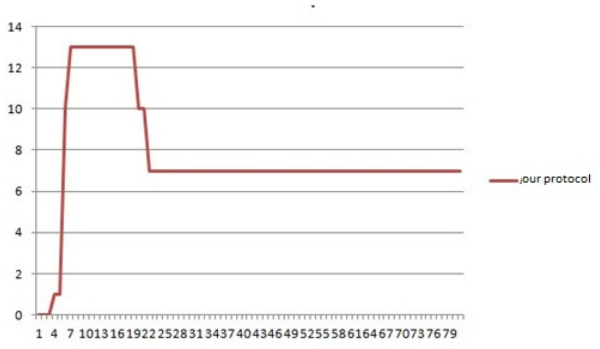


Figure 4: Convergence to target rate

Figure 4 shows the results of this experiment. From the figure 4, we can see that our protocol is very fast to converge. The convergence time is only 2 seconds for our protocol to converge to subscribe 15 layers without causing packet loss. It is also highly efficient in utilising the available bandwidth. The avour protocol ge throughput gained is approximately 271.93 $\pm$ 1.42 Kbps. The efficiency of network utilisation (E) is 0.9 $\pm$ 0.03.

Experiment I I: Intra-protocol Fairness Test

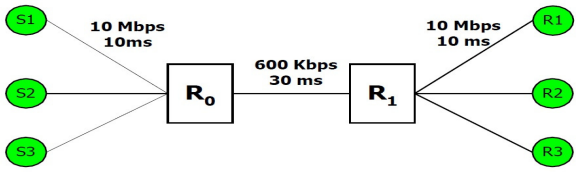
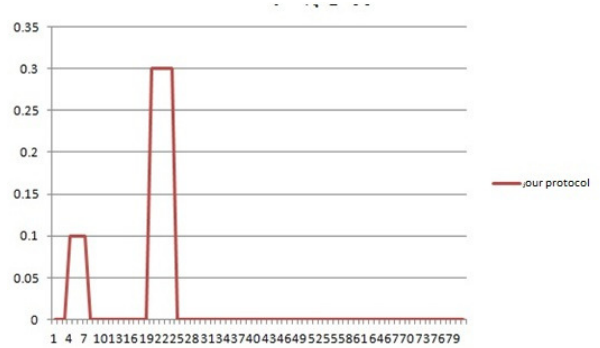


Figure 5: Topology of our protocol Experiment I I

The objective of this experiment set is to test the intra-protocol fairness of our protocol . The topology depicted in Figure 5 is used with three multicast sessions. Each one consists of one source and one sink. The bottleneck bandwidth is 600 Kbps, while each exterior link bandwidth is 10 Mbps. The simulation is run for 80 seconds. We start the first two sessions randomly during the first ten seconds, and start the third session at time 50 seconds.



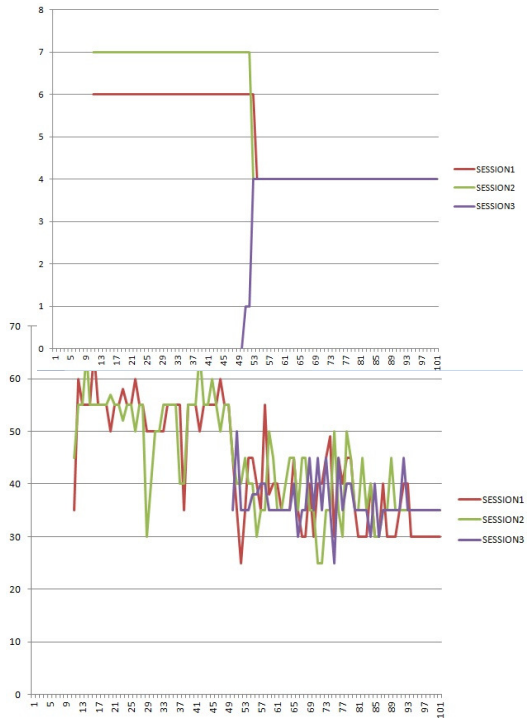


Figure 6: Intra-protocol fairness of 3 sessions

From Figure 6, we can see that our protocol demonstrates its intra-protocol fairness. When the first two sessions (Session 1 and 2) start, they can share the bottleneck bandwidth fairly (around 300 Kbps each) by subscribing 7 layers. After 50 seconds, when another our protocol session (Session 3) starts, three sessions of our protocol can still adjust the rate to be fair to each other to around 200 Kbps by each subscribing 4 layers.

5. Conclusions and Future Work

From the literature ([1], [2], [3], [4], [5]), there are several fundamental problems of RLM and RLC reported, such as slow convergence, unresponsiveness, TCP-unfriendliness, and high packet losses. This is because they have been designed using the join experiment. For RLC, although it is enhanced with the Burst Test technique, it is still prone to over-subscription and high packet losses and slow convergence. Furthermore, RLC was designed to be fair towards TCP whose RTT is one second only. It is aggressive towards TCP with RTT larger than one second, and submissive towards TCP with RTT smaller than one second.

We have extensively investigated FLID-DL is not TCP-friendly, slowly convergent, and has relatively high packet

loss rate. Compared with the experimental results of our protocol performs better.

However, PLM requires FQ at every router to enforce TCP-friendliness. This requirement is unfeasible, as some routers on the Internet still provide only drop-tail FIFO queuing. PLM without FQ can cause starvation of competing TCP connections. Hence, in terms of TCP friendliness, our protocol provides more feasible algorithms to achieve this goal.

We have developed a new experimental MR-MCC protocol (our protocol) by combining the following proven techniques:

1) The receiver-sided own-way packets available bandwidth estimation.
2) The TCP-friendly rate estimation of Padhye[11] .
3) The receiver-driven layered multicast framework of McCane[5] together with (1) Our new rate adaptation scheme, which is responsive and TCP-friendly
4) An innovative framework for the cooperation between the sender and receivers.

Our design goal is to provide MR-MCC with scalability, responsiveness, fast convergence, low packet loss rate, and fairness including TCP-friendliness. Our new experimental MR-MCC protocol has been successfully implemented and validated in the ns-2 network simulator. the performance comparison of our protocol with other MR-MCC proposals has been discussed and illustrated.

While we believe that our work has several claims of achievements, there would also be some weaknesses. In addition, several ideas have occurred during work on this research, and opened up several further avenues for exploration. The following subsections discuss some restrictions of this research and the issues that would be investigated as future work.

References

[1] J. W. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter, and W. Shaver, "FLID-DL: Congestion Control for Layered Multicast", In Proceedings of ACM NGC, pp. 71-82, Palo Alto, USA, November 2000.

[2] I. E. Khayat and G. Leduc, "A Stable and Flexible TCP-friendly Congestion Control Protocol for Layered Multicast Transmission", In Proceedings of the International Workshop IDMS, pp. 154-167, Lancaster, UK, October 2001.

[3] A. Legout and E. W. Biersack, "PLM: Fast Convergence for Cumulative Layered Multicast Transmission", In Proceedings of ACM SIGMETRICS, pp.13-22, Santa Clara, California, USA, June 2000.

[4] D. Sisalem and A. Wolisz, "MLDA: A TCP-friendly Congestion Control Scheme", In Proceedings of International Workshop on Quality of Service (IWQoS), pp. 65-74, Pittsburgh, PA, USA, June 2000.

[5] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven Layered Multicast", In Proceedings of ACM SIGCOMM, pp. 117-130, New York,USA, August 1996.

[6] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like Congestion Control for Layered Multicast Data Transfer", In Proceedings of IEEE INFOCOM, pp.996-1003, San Francisco, USA, April 1998.

[7] X. Li, S. Paul, and M. Ammar. Layered Video Multicast with Retransmission (LVMR):Evaluation of Hierarchical Rate Control. In Proceedings of IEEE INFOCOM, pages 1062–1072, March 1998.

[8] M. Jain and C. Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. IEEE/ACM Trans. Networking, 11:537–549, Aug. 2003.

[9] M. Jain, C. Dovrolis, "Pathload: a measurement tool for end-to-end available bandwidth", in Proc. Passive and Active Measurements (PAM) Workshop, Marseille, France, pp.14-25, 2002.

[10] J. D. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modelling TCP throughput: A Simple Model and its Empirical Validation", In Proceedings of ACM SIGCOMM, pp. 303-314, Vancouver, Canada, September 1998.

[11] J. Widmer and M. Handley, "TCP-friendly Multicast Congestion Control (TFMCC): Protocol Specification", IETF, Reliable Multicast Transport Working Group, Internet draft, July 2003, http://www.ietf.org/internetdrafts/draft-ietf-rmt-bb-tfmcc-01.txt. Work in progress. Expires January 2004.

[12] Sanjeev Mehrotra ,Jin Li Sudipta Sengupta , Manish Jain , Sayandeep Sen, "Hybrid Windowand Rate based Congestion Control for Delay Sensitive Application", IEEE Globecom 2010 proceedings.

[13] Gian Donato Colussi, " Equation-Based Layered *Multicast Congestion Control ", Helsinki November 2nd 2004 MS Thesis UNIVERSITY OF HELSINKI Department of Computer Science C-2004-71.*