On Receiver-Driven Layered Multicast Transmission

Jun Wei, Lixia Zhang
Computer Sciences Department, UCLA
4403 Boelter Hall, Los Angeles, CA 90095
E-mail: jun@cs.ucla.edu, lixia@cs.ucla.edu
April 17, 2000

Abstract

IP multicast allows data to be delivered to multiple receivers at the same time. Conventional unicast transmission control mechanism may not be directly applicable to IP multicast. Previous researches[6], [12] have proposed a layer-encoded multicast transmission scheme, which can efficiently send data to different receivers at different rates. In this report, we will present a thorough study on the throughput behavior of a layered multicast transmission with synchronized join experiment. Simulation experiments show that the throughput to different users can be adapted to the available bandwidth along different paths. The layered multicast transmission scheme is an open-loop control mechanism which can achieve intra-protocol fairness. Our studies also shows that an open-loop control protocol may not achieve inter-protocol fairness with close-loop control protocols, such as TCP.

I. Introduction

Multicast has provided an efficient way to deliver data to multiple receivers. In the meantime, it has been a great challenge for multicast senders to determine the transmission rate to multiple receivers. A unicast sender can adapt the transmission rate to the available bandwidth at the bottleneck point along the path to one receiver. A multicast session usually consists of multiple receivers. Multiple bottleneck points may occur in a multicast distribution tree, and the available bandwidth at different bottleneck points may differ from one another. Ideally, a multicast sender should be able to adapt to different bandwidth constraints at different bottleneck points. The layered multicast transmission mechanism is such a candidate.

McCanne, et. al. [6] have proposed a receiver-driven layered multicast transmission scheme (RLM). An RLM sender encodes a video stream into several sub-streams of different resolution levels. Each sub-stream is transmitted to a separate multicast address (sub-stream channel). Users with lower bandwidth connections to the sender do not subscribe to all the sub-stream channels in order not to overload the low bandwidth bottleneck routers. Therefore, different users subscribe to a different number of sub-stream channels and they receive the video with different resolution levels. The throughput to users differs from one another within the same multicast session.

In order to subscribe to an appropriate number of sub-streams, a RLM user conducts the join experiment to probe the available bandwidth. Each user starts by subscribing to the lowest resolution sub-stream (the base layer). If successful, it proceeds to the next higher layer, one by one. Whenever it detects a packet loss, the user drops the layer it is seeking. At some later time, it tries the join experiment again. A user backs off its join experiment after each failed attempt to probe the available bandwidth. Eventually a user subscribes to the appropriate number of sub-streams. The throughput to a user is the aggregate transmission rates of all the sub-streams that it has subscribed.

RLM requires that each RLM user conduct the join experiment independently. A new problem, called join experiment interference, emerges when several neighbor users try the join experiment for different sub-streams at the same time. A join experiment for a higher-level sub-stream interferes with a join experiment for a lower-level sub-stream. The packet loss caused by the join experiment for the higher-level sub-stream makes all users to back off. Some of these users may have been able to successfully subscribe to the lower-level sub-stream if they have not been interfered. McCanne, et. al. have introduced a solution, called a shared join experiment. Users are required to notify their neighbors of any ongoing join experiments. If a failed join experiment happens, the user trying for a lower-level sub-stream is allowed more chances to try again than the user trying for a higher-level sub-stream.

Vicisano et. al. [12] have proposed another solution to the join experiment interference problem. Instead of conducting the join experiment independently, users are only allowed to try the join experiment after receiving a special packet, called *synchronized point*, from the sub-stream of the base layer. The synchronized points for higher-level sub-stream layers are placed farther apart than the synchronized points for the lower-level sub-stream layers. After a failed join experiment, a user trying for a high level sub-stream layer has to wait for a longer time than the neighbor users trying for a lower-level sub-stream layers.

The synchronized join experiment can effectively prevent a higher-level join experiment from interfering with a lower-level join experiment.

The sender marks appropriate base layer packets to indicate the synchronized points for the different layers. The base layer is transmitted at a very low rate so that all users are able to subscribe to the base layer. Each user waits for the synchronized point for the second layer to subscribe to the second layer, and so on. Upon detecting a packet loss, a user should drop the current highest layer. The joining algorithm is summarized as follows:

- join a layer after the synchronized point of the interested layer.
- drop a layer upon detecting a packet loss.

Users drop a layer upon detecting a packet loss. However, the dropping of a problematic layer does not take effect until the underneath multicast routing tree is pruned. Therefore, a user continues to see packet losses for some time period after it has dropped a problematic layer. Vicisano et. al. [12] has suggested to wait for the *deaf period* (t_{deaf}) after a packet loss. A user does not response to further packet loss(es) during the deaf period. All losses that have happened within the deaf period are counted as one *loss event*.

Figure I shows a layered multicast session with synchronized join experiment. Four users A, B, C, and D are behind the same bottleneck point. They join the session at different times. The available bandwidth of the bottleneck point falls in between layer 2 and layer 3. User A joins the session first. By the time user B joins, user A is already trying for layer 2. The join

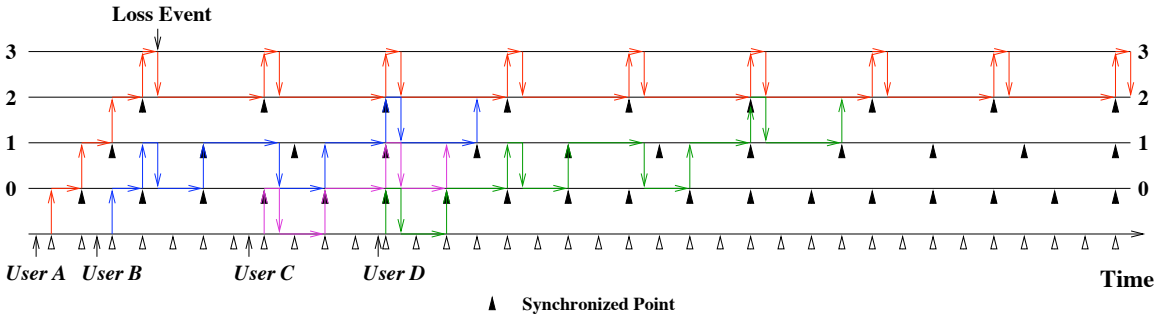

Fig. 1. At steady state, the subscription level of four users oscillated between layer 2 and 3. The time axis is on the scale of the synchronized point.

experiment by user A causes the first loss event. Both user A and B drop the current layer to layer 2 and layer 0, respectively. The join experiment by user A (for layer 3) interferes with the join experiment of user B (for layer 1). However, user B has more chances to try again for layer 1 than user A has for layer 3. Eventually, users B, C, and D catch up with user A to reach the steady state. The subscription level of all four users oscillate between layer 2 and layer 3.

Vicisano, et. al. [12] have shown that the average bandwidth of layered transmission with synchronized join experiment is inversely proportional to the square root of the loss event rate. However, their results only apply to an specific exponential layer structure. In this report, we present the analytical analyses to provide guidance for any data layer structures.

For the remaining of this report, we present analytical studies on throughput for layered multicast transmission with synchronized join experiment in section II. The studies provide the guidance on placing synchronization points for any arbitrary layer structures in order to achieve TCP-like steady-state throughput. In section III, we discuss the protocol behavior. In section IV, we investigate the impact of several essential parameters on the throughput by way of simulation experiments. We conclude our studies in section VI.

II. Throughput Analyses

Suppose the sender transmits the data layer i at the rate of $R_i (i = 0, 1, 2, ...)$. At steady state, when a user's subscription level is at layer i, its receiving rate, B_i, is the sum of the transmission rates of all the layers it has subscribed to (see Figure 2) and:

$$B_i = \sum_{j=0}^{j=i} R_j \tag{1}$$

The transmission rate of the base layer is R_0 and $B_0 = R_0$. Let τ_i denote the time period between two synchronized points for layer i. τ_i is also called as the *synchronization timer* for layer i.

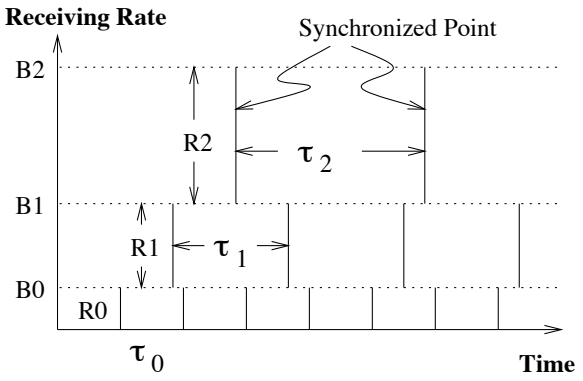

Fig. 2. The receiving rate B_i and the synchronization timer τ_i.

A. Average Throughput at the Steady-State

At steady-state, a user's subscription level oscillates between layer i and layer $(i+1)$ as shown in Figure 3. Let t_{i+1} denote the

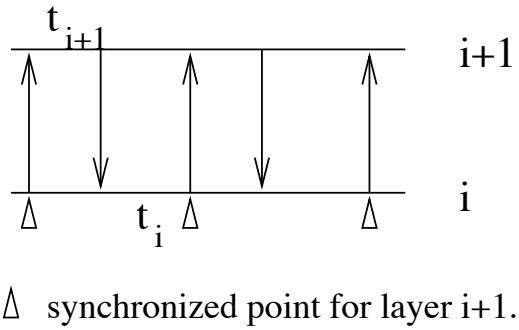

Fig. 3. At steady state, the subscription level of a user oscillated between layer i and $i+1$.

time period that a user spends on layer $(i+1)$ at steady state. t_{i+1} starts at the instant when the user successfully joins layer $(i+1)$ and extents to the instant when the user drops back to layer i upon detecting a loss event. The length of t_{i+1} is determined by the available bandwidth and the buffer size of the outgoing interface at the bottleneck link. The closer the available bandwidth matches B_{i+1}, the longer a user can stay on layer $i+1$. The sooner the outgoing buffer is filled up, the sooner a user experiences a loss event, and therefore, the shorter it can stay on layer $i+1$. Let α denote the ratio of t_{i+1} to the synchronization timer τ_{i+1}:

$$\alpha = \frac{t_{i+1}}{\tau_{i+1}} \tag{2}$$

Similarly, let t_i denote the time period that a user spends on layer i at steady state. t_i starts at the instant when the user drops from layer $(i+1)$ and extents to the instant before the user starts the next join experiment for layer $(i+1)$. Note that $t_{i+1} + t_i = \tau_{i+1}$.

Let N_{i+1} denote the number of the packets received during t_{i+1}, and N_i denote the number of the packets received during t_i:

$$\begin{aligned} N_{i+1} &= t_{i+1} B_{i+1} \\ N_i &= t_i B_i \end{aligned} \tag{3}$$

At steady state, the average throughput $B_{average}(i)$ of any user oscillates between layer i and layer $(i+1)$,

$$B_{average}(i) = \frac{N_{i+1} + N_i}{t_{i+1} + t_i} \tag{4}$$

Since $t_{i+1} + t_i = \tau_{i+1}$ and

$$N_{i+1} + N_i = t_{i+1} B_{i+1} + t_i B_i = \Big[\frac{B_{i+1}}{B_i} t_{i+1} + t_i\Big] B_i = \Big[\frac{B_{i+1}}{B_i}\alpha + (1-\alpha)\Big]\tau_{i+1} B_i \tag{5}$$

Therefore, the average throughput $B_{average}(i)$ can be rewritten as:

$$B_{average}(i) = \frac{N_{i+1}+N_i}{t_{i+1}+t_i} = \Big[\frac{B_{i+1}}{B_i}\alpha + (1-\alpha)\Big]B_i = \alpha B_{i+1} + (1-\alpha)B_i \tag{6}$$

Equation (6) is self explanatory. At steady state, a user spends a percent of time, α, at layer $(i+1)$ and the remaining percent of time, $(1-\alpha)$, at layer i. α can be obtained from the steady-state average throughput $B_{average}(i)$ as follows:

$$\alpha = \frac{B_{average}(i) - B_i}{B_{i+1} - B_i} \tag{7}$$

B. Loss Event Rate and Steady State Throughput

In this section, we show the relationship between the loss event rate and the steady-state throughput. Let p denote the loss event rate (section I) at steady state. Let s denote the packet size, assuming the packet sizes of each layer are equal. Suppose the transmission rates and the synchronization timers are chosen so that only one loss event will occur every τ_i as shown in Figure 3. The loss event rate p is

$$p = \frac{s}{N_{i+1}+N_i}, \quad \text{or} \quad \frac{s}{p} = N_{i+1} + N_i \tag{8}$$

The average receiving rate, $B_{average}(i)$, can be written in terms of loss event rate p:

$$B_{average}(i) = \frac{N_{i+1}+N_i}{t_{i+1}+t_i} = \frac{s}{p}\cdot\frac{1}{\tau_{i+1}} \tag{9}$$

Equation (5) can be expressed as:

$$\begin{aligned} N_{i+1}+N_i &= \Big[\frac{B_{i+1}}{B_i}t_{i+1} + t_i\Big]B_i \\ &= \Big[\frac{B_{i+1}}{B_i}\alpha + (1-\alpha)\Big]\frac{\tau_{i+1}}{\tau_i}\cdot\tau_i B_i \\ &= \Big[\frac{B_{i+1}}{B_i}\alpha + (1-\alpha)\Big]\frac{\tau_{i+1}}{\tau_i}\cdot\Big(\underbrace{\frac{\tau_i}{\tau_{i-1}}\cdots\frac{\tau_1}{\tau_0}}\Big)\cdot\Big(\underbrace{\frac{B_i}{B_{i-1}}\cdots\frac{B_1}{B_0}}\Big)\cdot\tau_0 B_0 \end{aligned} \tag{10}$$

Combining equation (5) with equation (8):

$$\frac{s}{p} = \Big[\frac{B_{i+1}}{B_i}\alpha + (1-\alpha)\Big]\frac{\tau_{i+1}}{\tau_i}\cdot\Big(\underbrace{\frac{\tau_i}{\tau_{i-1}}\cdots\frac{\tau_1}{\tau_0}}\Big)\cdot\Big(\underbrace{\frac{B_i}{B_{i-1}}\cdots\frac{B_1}{B_0}}\Big)\cdot\tau_0 B_0 \tag{11}$$

Equation (11) shows the relationship of the loss event rate p, the data layer structure B_i, and the synchronization timer τ_i. The loss event behavior is determined by two major factors. One is the sender who determines the data layer structure (B_i) and the synchronization timer (τ_i). The second factor is the network traffic load condition which can be represented by α (see equation 7).

C. TCP-like Steady-State Throughput

In this section, we show that the throughput of a layered multicast flow is TCP-like.
The sender can set the synchronization timer τ_i as following:

$$\frac{\tau_j}{\tau_{j-1}} = \frac{B_j}{B_{j-1}} = K_j, \quad (j = 1, 2, \ldots, i), \tag{12}$$

Equation (11) becomes:

$$\begin{aligned} \frac{s}{p} &= [K_{i+1}\alpha + (1-\alpha)]K_{i+1}\cdot(\underbrace{K_i\cdots K_1})\cdot(\underbrace{K_i\cdots K_1})\cdot\tau_0 B_0 \\ &= [K_{i+1}\alpha + (1-\alpha)]K_{i+1}\cdot(\underbrace{K_i\cdots K_1})^2\cdot\tau_0 B_0 \end{aligned} \tag{13}$$

or,

$$\underbrace{(K_i \cdots K_1)}^{2} = \frac{s}{p} \cdot \frac{1}{[K_{i+1}\alpha + (1-\alpha)]K_{i+1} \cdot \tau_0 B_0} \tag{14}$$

τ_{i+1} can be rewritten as:

$$\tau_{i+1} = \frac{\tau_{i+1}}{\tau_i} \cdot \underbrace{\left(\frac{\tau_i}{\tau_{i-1}} \cdots \frac{\tau_1}{\tau_0}\right)} \cdot \tau_0 = K_{i+1} \cdot \underbrace{(K_i \cdots K_1)} \cdot \tau_0 \tag{15}$$

By placing the synchronized points according to equation (12), the steady-state throughput, $B_{average}(i)$, to a layered multicast user becomes:

$$\begin{aligned} B_{average}(i) &= \frac{s}{p} \cdot \frac{1}{\tau_{i+1}} \\ &= \underbrace{\sqrt{\frac{[(K_{i+1}-1)\alpha + 1] \cdot m}{K_{i+1}}}} \cdot \frac{1}{\sqrt{p}} \cdot \frac{s}{\tau_0} \\ &= K'(i) \cdot \frac{1}{\sqrt{p}} \cdot \frac{s}{\tau_0}. \end{aligned} \tag{16}$$

where,

$$K'(i) = \sqrt{\frac{[(K_{i+1}-1)\alpha + 1] \cdot m}{K_{i+1}}} \tag{17}$$

and $B_0 = \frac{s}{t_0} = \frac{ms}{\tau_0}$. $K'(i)$ is a function of $K_{i+1} = B_{i+1}/B_i$, m and α. Except for α, the other parameters are determined by the sender. α is determined by the outgoing buffer size and the available bandwidth at the bottleneck link. $K'(i)$ is a constant along a specific path. Users behind the same bottleneck link have equal $K'(i)$.

The steady-state throughput (equation 16) is similar to the steady-state throughput of a TCP connection [4], [5].

$$B_{average} = c \cdot \frac{1}{\sqrt{p}} \cdot \frac{MTU}{RTT} \tag{18}$$

MTU represents the TCP packet size and RTT is the round-trip time. A layered multicast flow responds to the loss event rate in a manner similar to a TCP flow.

In order to achieve the steady-state throughput (equation 16), the sender should place the synchronized points according to equation (12). Next, we present the placement of synchronized points for the two data layer structures.

1. **Exponential Data Layers:**

One example of data layer structure is the exponential layer:

$$B_j = 2^j B_0, \quad j = 0, 1, 2, \ldots, i. \tag{19}$$

The sender should place the synchronized points as follows:

$$\frac{\tau_j}{\tau_{j-1}} = \frac{B_j}{B_{j-1}} = K_j = 2, \quad j = 1, 2, \ldots, i. \tag{20}$$

or,

$$\tau_j = 2^j \tau_0, \quad j = 0, 1, 2, \ldots, i. \tag{21}$$

The steady-state throughput, $B_{average}(i)$, (equation 16) become:

$$B_{average}(i) = \sqrt{\frac{(\alpha+1) \cdot m}{2}} \cdot \frac{1}{\sqrt{p}} \cdot \frac{s}{\tau_0} \tag{22}$$

2. **Linear Data Layers:**

Another example of data layer structure is the linear layer:

$$B_j = (j+1)B_0, \quad j = 0, 1, 2, \ldots, i. \tag{23}$$

The sender can place the synchronized points as follows:

$$\frac{\tau_j}{\tau_{j-1}} = \frac{B_j}{B_{j-1}} = K_j = \frac{j+1}{j}, \quad j = 1, 2, \ldots, i. \tag{24}$$

or,

$$\tau_j = (j+1)\tau_0, \quad j = 0, 1, 2, \ldots, i. \tag{25}$$

The steady-state throughput, $B_{average}(i)$, (equation 16) becomes:

$$B_{average}(i) = \sqrt{\frac{(\alpha+i)\cdot m}{1+i}} \cdot \frac{1}{\sqrt{p}} \cdot \frac{s}{\tau_0} \tag{26}$$

III. Protocol Behaviors

In this section, we discuss the factors that affect the layered multicast behavior, such as the response time, the steady-state behavior, starting behavior, fairness, and stability, etc. We compare the layered multicast flows with TCP flows. We compare two data layer structures and their suitability in practical uses. At the end, we present the synchronized join experiment with exponential back off.

A. Protocol Behavior

Response time

In a TCP connection, an explicit control loop (by way of acknowledgement) is set up between the sender and the receiver. The acknowledgement from the receiver allows the sender to respond to the congestion within one round-trip time. The layered multicast transmission is designed to avoid explicit control loop(s) between the sender and multiple users. The sender does not collect the acknowledge and respond to the congestion. In contrast, the multicast users detect the congestion and respond to it by unsubscribing from the higher-level data layers. The response does not take effect until the multicast distribution tree is pruned. The response time is the sum of the one-way delay and the prune delay.

Steady State Behavior

A TCP sender responds to the congestion by cutting the window size by half and then increasing the window size linearly by one packet per round-trip time. The TCP throughput then increases linearly until another packet loss. The instantaneous TCP throughput oscillates at steady state. Similarly, the multicast users drop an entire data layer upon a loss event. They then wait for the next synchronization point to join again. The steady state throughput is a step function oscillating between two layers.

Starting Behavior

At the beginning, a TCP starts with the window size of one and doubles the window size every round-trip time. The throughput increases exponentially (TCP's slow start). A TCP flow can reach its steady state exponentially within a few round-trip times. On the other hand, the throughput of a layered multicast flow increases linearly as a step function. The throughput increases in a linear manner (see Figure 4). A layered multicast flow reaches its steady state slowly and steadily.

Fairness

Intra-Protocol Fairness: When two TCP flows share a common bottleneck link, the TCP flow with a shorter round-trip time utilizes more bandwidth than the TCP flow with a longer round-trip time (see equation (18)). On the other hand, the steady-state throughput of the layered multicast flow (see equation (16)) does not show any dependency on the round-trip time. Rather, it dependents on a few parameters determined by the sender, such as the base layer rate B_0 and the synchronization timer of the base layer τ_0. When two multicast flows share the same bottleneck link, they can evenly share the bandwidth if they have the same B_0 and τ_0.

Inter-Protocol Fairness[1]. When a layered multicast flow shares a link with a TCP flow, they both respond to the loss event the same way, except for a different factor. The steady-state throughput is inversely proportional to the square root ofthe loss event

[1]More accurate modeling of TCP throughput is in [8]. We do not expect the layered multicast to be exactly TCP-like. The design goal is to be less aggressive than a TCP connection under the same conditions.

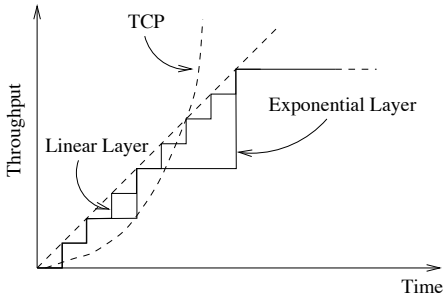

Fig. 4. The starting behavior.

rate. A multicast distribution tree usually contains multiple bottleneck links. The layered multicast transmission allows different throughput along different bottleneck link. The throughput along each bottleneck links follows the inverse square root rule. The layered multicast flow does not outrun any TCP flows sharing any of the bottleneck links.

Stability

Both TCP and layered multicast transmission exhibit throughput oscillating behavior at steady state. The oscillating behavior is an evidence of the stable state that the throughput can increase or decrease when it has reached the lower or upper bound.

B. Data Layer Structure

In this section, we compare two data layer structures: the linear layer structure and the exponential layer structure.

For a fair comparison, the base layer rate, B_0, and the synchronization timer, τ_0, are set to the same values for both the exponential layer structure and the linear layer structure. At steady state, the throughput of the linear layer oscillates within a

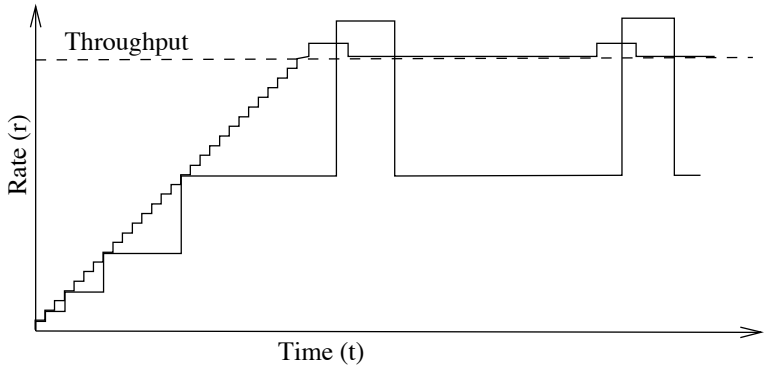

Fig. 5. The throughput of linear layer v.s. exponential layer.

narrower range than the throughput of the exponential layer. It takes the same amount of time, for the multicast flows with both the linear layer and the exponential layer, to reach the steady state as shown in Figure 5. However, the increase step in throughput is much larger for the exponential layer than for linear layer. Therefore, the layered multicast flow with the exponential layer introduces more bursty traffic into network than the layered multicast flow with the linear layer.

The exponential layer requires a fewer number of layers to cover the same range of available bandwidth than the linear layer. For example, it takes 6 layers of exponential layers to cover the bandwidth range from 32 kbps to 1,024 kbps, and it takes 22 linear layers if both use the same base layer rate B_0 of 32 kbps.

Figure 5 shows that the linear layer structure allows users to adjust to the available bandwidth with finer granularity than the exponential layer structure. However, the price of the finer granularity is the number of the multicast addresses to be used. The linear layer structure requires more multicast addresses than the exponential layer structure for the same bandwidth range.

C. Exponential Back-off after Dropping a Layer

As mentioned previously in section II, a user's subscription level oscillates between two adjacent layers at steady state. To avoid oscillation, the damping technique can be used. A user backs off the join experiment after a failed trial(see Figure 6). For example, the user passes one synchronized point after a failed join experiment (see Figure 6). It passes three synchronized points

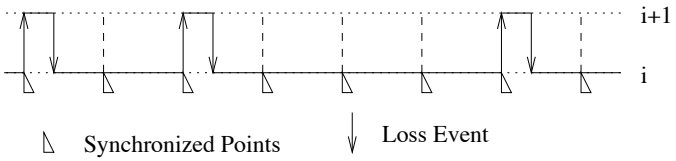

Fig. 6. Exponential back off after failed join experiment.

after the second failure, and so on. Eventually, the user settles at the layer just below the value predicted by equation (16).

$$\begin{aligned} B_{average}(i) &= \lim_{M\to\infty} \frac{\sum_{m=0}^{M} N_{i+1} + \sum_{m=0}^{M} 2^m N_i}{\sum_{m=0}^{M} 2^m \tau_i} \\ &= \lim_{M\to\infty} \frac{mN_{M+1}}{\sum_{m=0}^{M} 2^m \tau_i} + \frac{N_i}{\tau_i} \\ &= \frac{N_i}{\tau_i} \end{aligned} \tag{27}$$

IV. Simulation Experiments

In the experiments described in [13], it has been shown that users of the same multicast session is able to subscribe to different number of layers based on their individual connections to the sender. The receiving rate or the throughput of each user relates to the aggregate transmission rates of all the layers it has subscribed to. In this section, we study the throughput of the users behind the same bottleneck link through a simple topology configured as shown in Figure 7. Specifically, we study the throughput behavior of a layered multicast flow with the present of competing flows using other transport protocol, e.g. TCP. To understand whether the encoded layer structure would affect the throughput behaviors, both the linear layer structure and the exponential layer structure are implemented for the layered multicast transmission mechanism in the NS simulator [7].

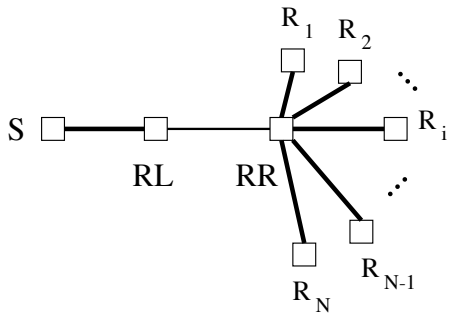

Fig. 7. The topology for layered multicast experiment.

Multiple TCP connections are set up between S and R_i ($i = 1, 2, ..., N$). The round-trip times (RTT) of the TCP flows, in the absence of the queueing delay, vary from 44 to $(44 + 2N/3)$ ms (not to exceed 76 ms). Multiple layered multicast flows are also set up between S and R_i. The sender of all multicast flows is at S. Each multicast flow has two users randomly chosen from R_1 through R_N. All TCP and multicast flows shared the same bottleneck link between RL and RR. Another four low-bandwidth TCP flows are configured in the reverse direction from RR to RL to avoid the phase effects of TCP flows.

The bottleneck bandwidth is 15 Mbps. The outgoing buffer size at the router RR is 40 packets. All TCP flows and layered multicast flows use the same packet size of 1000 bytes. In the absence of queueing delay, the delay bandwidth product ranges from 82 to 142 packets. The starting times of the individual flows are 0.1 second apart. The experiments run from 0 to 30 seconds, and the measurements are taken from 15 to 30 second.

A. The Synchronization Timer τ_0

According to equation 16, the steady-state throughput directly relates to the base layer synchronization timer τ_0. In this section, we study how the synchronization timer τ_0 affects the average throughput.

Four layered multicast flows (MCC) and four TCP flows are configured to share the common bottleneck link between RL and RR. The average throughput achieved by each flow is measured over 15 seconds. The base layer rate B_0 is fixed at 400kbps

and the linear layer structure is used. The average throughput is measured at the different values of the synchronization timer τ_0. Figure 8 shows the mean throughput v.s. the synchronization timer τ_0. The bandwidth utilization of a layered multicast flow

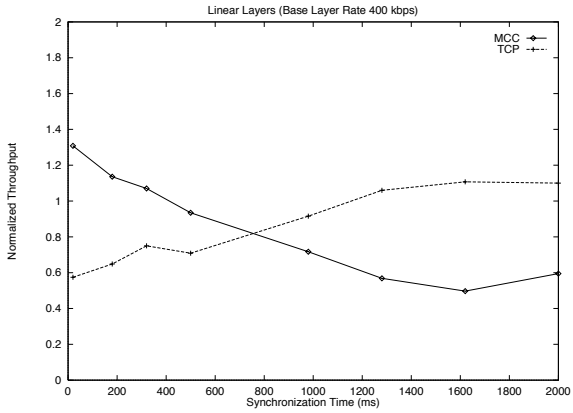

Fig. 8. The mean throughput v.s. the synchronization timer τ_0. The base layer rate B_0 is 400kbps. The TCP round-trip times are between 44ms to 47ms.

declines with the synchronization timer τ_0 with the present of competing TCP traffic. The larger the synchronization timer is, the less frequently a user can increase the subscription level. Increasing the synchronization timer makes a layered multicast flow less aggressive.

B. The Base Layer Transmission Rate B_0

The base layer transmission rate B_0 determines the data rate injected into the network by a layered multicast sender. Equation 16 shows that the steady-state throughput is independent with the base layer transmission rate B_0. In this section, we study whether B_0 would affect the average throughput of the layered multicast flows and the total bandwidth utilization at the bottleneck point. The experiment setup is the same as in section IV-A. The synchronization timer τ_0 is fixed and the average throughput of each flow is measured at different values of the base layer transmission rate B_0. Figure 9 shows the results from one of the experiments with the linear layer. The mean normalized throughput of four multicast flows v.s. the base layer rate B_0 is plotted in Figure 9. The mean throughput of four TCP flows and the total bandwidth utilization (in the units of 2.0) of the bottleneck link are also plotted in Figure 9 for reference.

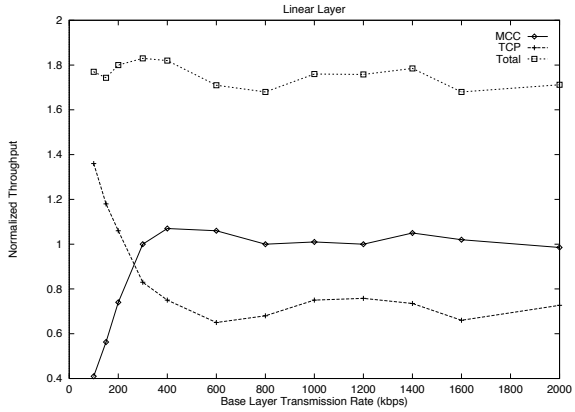

Fig. 9. The mean normalized throughput v.s. the base layer rate B_0. The synchronization timer τ_0 is 80ms.

Figure 9 shows a threshold value of the base layer rate B_0. When B_0 is above the threshold, the average throughput of the layered multicast flow is independent of B_0. When B_0 is below the threshold, the throughput of a layered multicast flow is substantially low. This is because the multicast users can only increase the subscription level as fast as one layer for every τ_0 (80ms). In this experiment, 80ms is approximately twice as much of the round-trip time of TCP flows (44ms). The TCP flows are much more aggressive than the layered multicast flows when B_0 is below the threshold.

We have run similar experiments at the synchronization timer of 160ms and 250ms. The normalized throughput v.s. the base layer rate B_0 is plotted in Figure 10. The threshold of B_0 increases with the synchronization timer τ_0. This is because a large τ_0

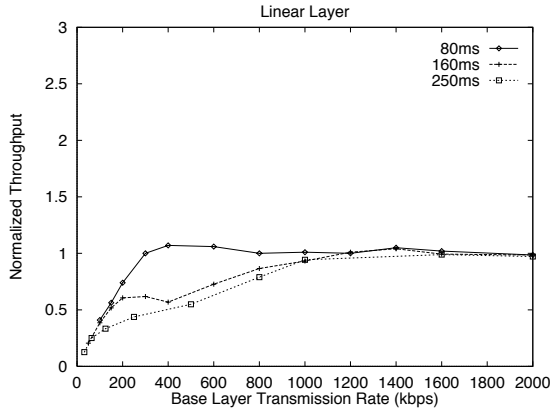

Fig. 10. The mean normalized throughput v.s. B_0 at different τ_0.

usually results in the slow increase of a user's subscription level. We also ran experiments with a different number of flows and with exponential layer structure. The threshold of B_0 varies with the number of concurrent flows and the layer structure.

C. Linear Layer v.s. Exponential Layer

In this section, we compare two layer structures, namely, the linear layer and the exponential layer. The same configuration as in section IV-B is also used. We have run the identical experiments with the linear layer and with the exponential layer. The mean normalized throughput for both sets of the experiments are shown in Figure 11 (the TCP flows are not shown). The total bottleneck link utilization (in the units of 2.0) of both sets of the experiments are also shown in Figure 11. The synchronization

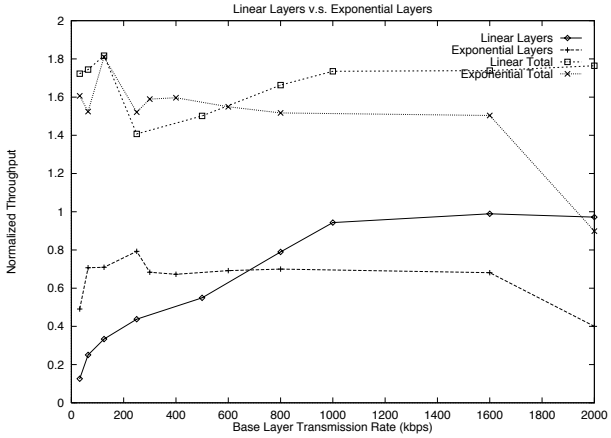

Fig. 11. The mean normalized throughput v.s. the base layer rate B_0 for both the linear layer and the exponential layer. The total bottleneck link utilization is also plotted. The synchronization timer τ_0 is $250ms$.

timer τ_0 is 250 ms for both sets of the experiments. Figure 11 shows that the layered multicast flows with the exponential layer is more aggressive than the flows with the linear layer at small B_0. At large B_0, flows with the linear layer utilize the link bandwidth more efficiently than flows with the exponential layer.

D. Intra-Protocol Fairness

In this section, we study the intra-protocol fairness for different number of multicast flows. The configuration in Figure 7 is used for $N = 2, 4, 8, 16, 24,$ and 32. The same number of TCP flows (N) run from RL to RR as the background traffic. For each N, B_0 is 32kbps and τ_0 is 250ms. The average throughput of each multicast flow (MCC) are shown in Figure 12. The mean value of all N MCC flows are also plotted for each N. For the linear layer encoding structure, all N ($N < 32$) flows achieved the same average throughput for most N. When all N MCC flows use the same τ_0 and B_0, they all achieve the same throughput, regardless of the number of concurrent MCC flows. Similarily, the average throughput achieved by each MCC flows with exponentially encoded layers are within a narrow range for small number of MCC flows. Figure 12 shows that the layered multicast flows are fair with each other when they use the same τ_0 and B_0.

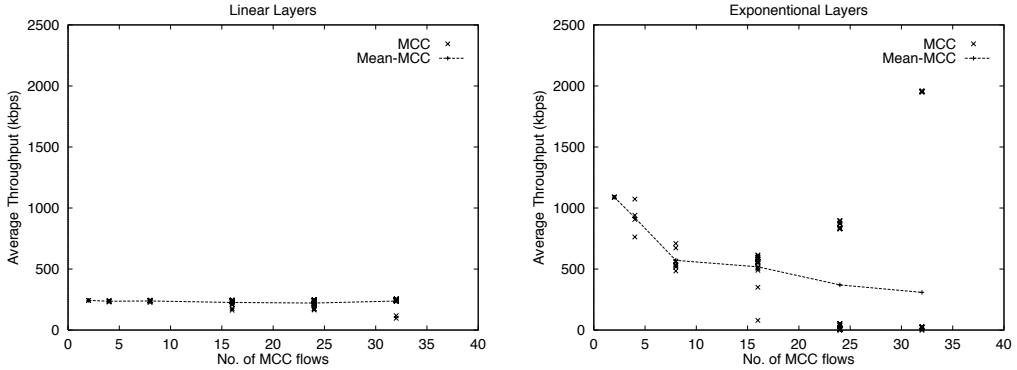

Fig. 12. The intra-protocol fairness. B_0 = 32 kbps, τ_0 = 250 ms.

When $N = 32$ for linear layer structure, two out of thirty-two flows resulted in substantially lower throughput than others. Also when $N = 24$ or 32 for exponential layer structure, some flows achieved much lower bandwidth share than others. The bandwidth unfairness disappears when different τ_0 and B_0 are used, namely larger τ_0 and smaller B_0. This implies that one set of the fixed value of τ_0 and B_0 may not suit for different number of concurrent layered multicast flows. In other words, a layered multicast sender need to pick the correct values of τ_0 and B_0 in order to be fair with other layered multicast flows.

Figure 12 also shows that the layered multicast flows with exponential layers achieved higher throughput than the flows with linear layers under the same situations, i.e. with the same τ_0 and B_0 and the same number of competing TCP flows in the backgroud.

E. Inter-Protocol Fairness

In this section, we use the same experiment setup as in the previous section to study the inter-protocol fairness. The average throughput of each flow is normalized to the $1/2N$th of the total link bandwidth at the bottleneck point. The normalized throughput for TCP flows are added in Figure 13. τ_0 and B_0 are adjusted for small N and large N. Table I lists the synchronization timer τ_0 and the base layer transmission rate B_0 used in the experiments.

N	1	2	4	8	16	24	32	40	48
$\tau_0(ms)$	160	160	160	160	160	160	250	333	500
$B_0(kbps)$	1600	800	500	200	100	50	32	24	16

TABLE I

THE BASE LAYER RATE B_0 AND THE SYNCHRONIZATION TIMER τ_0 USED IN THE INTER-PROTOCOL FAIRNESS EXPERIMENT.

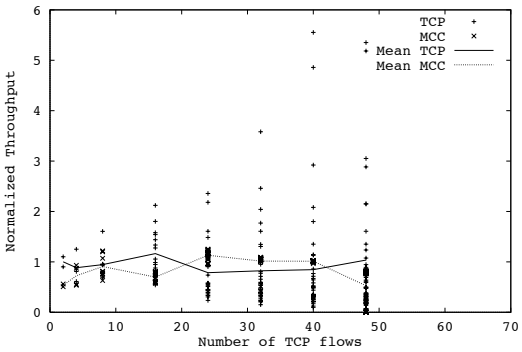

Fig. 13. The inter-protocol fairness experiment.

The average throughput of a TCP flow depends on the round-trip delay. In experiments, the round-trip delay of all TCP flows

ranges from 44ms to 76ms. The average throughput of the TCP flows varies within a wider range. On the other hand, the layered multicast flows do not show any dependency on the round-trip delay. Although the round-trip delays from the sender to various users vary within the same range as the TCP flows, the average throughput of all layered multicast flows varies within much narrower ranges as shown in Figure 13.

F. Discussion

Figure 13 shows a layered multicast flow can achieve a throughput compatible to the throughput of the TCP flows when the correct value of τ_0 and B_0 are used. To achieve the TCP-compatible throughput, a layered multicast flow can adjust τ_0 and B_0 based on equation 16 and 18. For example,

$$\begin{gathered} K'(i) \cdot \frac{1}{\sqrt{p}} \cdot \frac{s}{\tau_0} = c \cdot \frac{1}{\sqrt{p}} \cdot \frac{MTU}{RTT} \\ or \\ K'(i) \cdot \frac{1}{\tau_0} = c \cdot \frac{1}{RTT} \end{gathered} \tag{28}$$

where K' relates to B_0 and α as shown in equation 17 and α can be measured by users as indicated by equation 7.

The experiments from the last two sections suggest that the layered multicast flows should have additional means to adjust B_0 and τ_0 in order to make them TCP-compatible or TCP-friendly. Both the sender and the users can do such adjustments. If the sender were to adjust B_0 and τ_0, it would require some form of feedback mechanism, for example, feedback from the users to measure the round-trip delay, etc. The feedback from the users closes the control loop between the sender and the users.

V. Related Work

In this section, we review several layered multicast transmission schemes and a single-rated transmission scheme for multicast data delivery.

Deering [3] first suggests that the layered transmission can be used for multicast delivery. Others [10] also described the similar scheme. McCanne, et. al. [6] presented a receiver-based layered multicast(RLM) scheme for multimedia real-time applications. The sender encodes the real-time video stream into several sub-streams of different resolution levels and transmits each sub-stream to a separate multicast address or channel. Different users subscribe to a different number of sub-streams. Each user conducts the join experiment independently to determine the appropriate subscription level. Users with high bandwidth connections subscribe to more sub-streams than the users with low bandwidth connections. As a result, the throughput to heterogeneous users varies from one another within the same multicast delivery session.

However, independent join experiment can cause the join experiment interference problem. RLM suggests the shared join experiment that requires each user notify others of any ongoing join experiment. Exponential layers are used in RLM. The throughput behavior of RLM flows remains for further study [6].

Vicisano, et. al. [12] proposes the synchronized join experiment to solve the join experiment inference problem. A join experiment is only allowed at the synchronized points. The sender marks the base layer packets as the synchronized points for higher layers. The synchronized points of a higher layer are placed further apart than those of a lower layer. After a join experiment fails, the users trying at a lower layer have more chances to try again than users trying at a higher layer.

However, the study in [12] only simulated the exponential data layer structure. We present thorough throughput analyses and provide the guideline for placing the synchronized points for other data layer structures. We conduct extensive simulation experiments to study the throughput behavior.

Thierry, et. al. [2], [11] proposes a layered transmission scheme for RTP-based [9] audio applications. Instead of probing the available bandwidth by way of the join experiment, users monitor the packet loss rate from RTCP reports. Based on the equation 18, users calculate the TCP-friendly bandwidth share at that particular packet loss rate. The result is used as the upper limit for the user to determine the maximum number of layers to be subscribed to.

However, this approach may cause some users to underestimate available bandwidth under certain circumstance. Suppose two users are both behind the same bottleneck link, one is further from the sender than the other. Due to the common bottleneck link, both users experience the same packet loss rate. The TCP-friendly bandwidth share calculated by two users are different because the round-trip times (RTT) from the sender to them are different. The TCP-friendly bandwidth share is inversely proportional to RTT [4], [5] (see also equation (18)). Therefore, the further user subscribe to fewer number of layers than the nearer user even though they both can subscribe to the same number of layers.

Bhattacharyya, et. al. [1] proposes a layered multicast transmission scheme for bulk data transfer. The data is divided into several parts and carefully arranged to be continuously sent to separate multicast addresses or channels. Users subscribe to an appropriate number of layers. Similar to RLM, users with low bandwidth connections subscribe to fewer channels than users with

high bandwidth connections. The data layer is so arranged that any user is able to receive all the data regardless what channel and how many channels it subscribes to. The drawback of this scheme is that the sender has to continuously feed the data channel. The same data may be sent several times over different channels to make sure that users subscribing to different channels can all receive one copy of the data.

VI. Summary

In this report, we have thoroughly analyzed the layered multicast transmission with the synchronized join experiment. We have provided the guideline layered multicast transmission with any layer structure to adapt to different bandwidth constraints by way of the synchronized layered multicast transmission. We have implemented the layered multicast transmission with both the linear layer structure and exponential layer structure in the NS simulator and have conducted extensive experiments to investigate the throughput behavior.

Our simulation experiments show that the base layer synchroniztion timer $\tau_0$and the base layer transmission rate B_0 are two essential protocol parameters. The steady-state throughput of a layered multicast flow is inversely proportional to the base layer synchronization timer τ_0. There is a critical threshold value of the base layer rate B_0. Below the threshold value, the bandwidth utilization of the layered multicast flows is much smaller compared with that of the competing TCP flows. With a very large value of the base layer rate, bandwidth unfairness may occur among multiple layered multicast flows and the total link utilization may become low as well. The throughput of the layered multicast flows is independent of the round-trip delays from the sender to any users. At lower base layer rate, the layered multicast flow with the exponential layer structure utilizes more bandwidth than the flow with the linear layer structure. However, at larger base layer rate, the exponential layer can cause bandwidth unfairness and low link utilization more easily than the linear layer structure. The linear layer structure is more favorable than the exponential layer structure when a large number of concurrent layered multicast flows are present.

The layered multicast transmission scheme is an open-loop control scheme. The steady-state throughput is able to adapt to the different bandwidth contraints within the same multicast session. In order to achieve a throughput compatible with a close-loop control scheme, such as TCP, additional means are required to adjust the base layer rate and the synchronization timer. Some form of feedback mechanisms will be required to close up the control loop from the sender to users. The clustering algorithm [13] will help to handle heterogenous feedback signals from multiple users.

References

[1] Supratik Bhattacharyya, James Kruose, Don Towsley, and Ramesh Nagarajan. Efficient rate-controlled bulk data transfer using multiple multicast groups. IEEE INFOCOM, April 1998.

[2] Jean-Chrysostome Bolot. Layered multicast data transfer. Viewgraph for Reliable Multicast Meeting of IRTF, September 1997. Obtained via: www.east.isi.edu/rm/bolot.ps.

[3] Stephen Deering. Internet multicast routing: State of the art and open researchissues. In *Multimedia Integrated Conferencing for Europe(MICE) Seminar atSwedish Institute of Computer Science*, Stockholm, October 1993.

[4] Sally Floyd. Connections with multiple congested gateways in packet-switched networks part 1: One way traffic. *Computer Communication Review*, 21(5):30–47, October 1991.

[5] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *Computer Communication Review*, 27(3), July 1997.

[6] Steven McCanne, Van Jacobson, and Martin Vetterli. Receiver-driven layered multicast. In *Proceedings of ACM SIGCOMM'96*, pages 117–130, Stanford, CA, August 1996.

[7] *UCB/LBNL/VINT Network Simulator, Version 2*. URL: http://www-mash.cs.berkeley.edu/ns.

[8] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling tcp throughput: A simple model and its empirical validation. In *Proceedings of ACM SIGCOMM'98*, Vancuvour, Canada, September 1998.

[9] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*, January 1996. Internet RFC 1889.

[10] Thierry Turletti and Jean-Chrysostome Bolot. Issues with multicast video distribution in heterogeneous packet networks. In *Proceedings of the Sixth International Workshop on Packet Video*, Portland, OR, September 1994.

[11] Thierry Turletti, Sacha Forsse Parisis, and Jean-Chrysostome Bolot. Experiments with a layered transmission scheme over the internet. Technical report, INRIA, Sophia Antipolis, November 1997. **U**RL: http://www.inria.fr.

[12] Lorenzo Vicisano, Luigi Rizzo, and Jon Crowcroft. Tcp-like congestion control for layered multicast data transfer. In *Proceedings of IEEE Infocom'98*, San Francisco, CA, April 1998.

[13] Jun Wei and Lixia Zhang. A helper based clustering algorithm for multicast data delivery. CSD-TR-200015, UCLA, 2000.