Fairness between layered multicast data transfer and TCP

by

Yu Hong Lin

(96609007G)

A Dissertation submitted in fulfillment

of the requirement for the degree of

Master of Science in Information Technology

Department of Computing

Hong Kong Polytechnic University 2001

Acknowledgments

I would like to thank my supervisor, Rocky Chang's continuous support, constant encouragement and enthusiasm. He devoted lots of him time in giving me valuable helpful opinions. I specially thank him for showing me the right way to do research.

I am also grateful that Thomas Wong, Eddie Wong, Ben Tam, my colleagues and friends, provided feedback on various parts of this work.

I gratefully acknowledge Alvin Chan, K. W. Hung and M.Z. Wang for participating in my dissertation defense.

Finally, I would like to dedicate this work to my wife, Monica Wong, and son, Fung Fung, who have always support me and give me meaning and wholeness.

Abstract

Steaming multimedia applications are becoming increasingly popular in the Internet. Among all transportation mechanisms, layered multicast is one of the best choices to deliver multicast multimedia data in the Internet. However, there is significant concern about the effects on co-existing between TCP and layered multicast. Lack of an effective and “TCP friendly” congestion control is the main barrier for the wide-range deployment of the layered multicast applications.

In this dissertation, I design and evaluate an end to end congestion control for layered multicast. I propose two new congestion controls for layered multicast – Middle Layer Dropping (MLD) and Entire Layer Dropping (ELD). To make MLD and ELD “TCP-like”, MLD and ELD simulate the behaviors of the TCP congestion windows. Moreover, MLD and ELD introduce a “router assistance” factor to improve the intra-fairness among the receivers. I also examine both MLD and ELD through simulations. The simulations reveal that both congestion controls do illustrate the TCP fairness properties.

Contents

1. Introduction

The introduction of IP multicast applications adds more functions to the Internet. With IP multicast, some unique and powerful applications and application services, which are impossible in unicast environment, can now become possible. Among all types of multicast applications, multimedia is a most rapid-growing area. It is believed that IP multicast multimedia applications will become more popular in the next few years. However developing the multicast multimedia applications is challenging and there are two main issues need to be addressed before the applications can be successfully deployed in the Internet.

1. *What is the impact of the new multicast traffic on TCP on the Internet (Inter fairness issue)?*

2. *How can a multicast group accomodate a large number of heterogeneous receivers, which have a wide range of available bandwidths and network conditions (Intra fairness issue)?*

1.1 Question 1 - TCP fairness issue

To avoid congestion, end systems are expected to be cooperative by reacting to congestion and adapting their transmission rates properly and promptly. Currently, the majority traffic in the Internet is best effort, TCP traffic. TCP uses an Additive Increase Multiplicative Decrease (AIMD) mechanism, in which the sending rate is controlled by a congestion window. The congestion window is halved for every window of data containing a packet drop and increased by roughly one packet per window of data otherwise. Similarly, IP multicast should have its congestion control algorithm. However IP multicast cannot simply adopt the TCP congestion control algorithm because of acknowledgement causing “implosion problem” in IP multicast. Due to the different congestion control algorithms between TCP and multicast, the network bandwidth may not be shared fairly between the competing TCP and multicast flow. Lack of an effective and “TCP friendly” congestion control is the main barrier for the wide-range deployment of multicast applications.

One of the possible solutions is to use the differential services for IP multicast. However, the differential or reservation services have not been widely used in the Internet nowadays. Even in the future, I believe there would still be a significant amount of traffic (TCP or IP multicast) transmitted under best-effort environment. As a result, a “TCP friendly” congestion control is still needed and crucial for the well-being of the Internet.

1.2 Question 2 – Heterogeneity of receivers

The heterogeneity of the receivers under an IP multicast session significantly complicates the problem of effective data transmission. One of the major problems in IP multicast is the sending rate the sender chooses. If the transmission rate is too high, it will cause a packet loss or even a congestion collapse, whereas low transmission rate will leave some receivers underutilized. This problem has been studied for many years and is still one of the active research areas in IP multicast. Actually, it is another fairness issue among the receivers in a multicast group.

Many papers have proposed different congestion control algorithms for IP multicast and basically those proposed algorithms could be classified into two categories:

- Single-rate with single multicast group – LTRC [8], MBFC [9]

- Multi-rate with multiple multicast groups- RLM [4], RLC [5]

Single-rate cannot perform well in terms of the level of heterogeneity among the receivers. It is because the sender can only adjust the sending rate to match the requirement of one or a portion of receivers but it cannot meet the conflicting requirements of multiple heterogeneous receivers.

Multi-rate permits multi-rate transmission. By using multi-rate, slow receiver can receive data at a slow rate while fast receiver can receive data at a fast rate. In general, multi-rate congestion control can perform well in a fair manner for a large multicast group with a large number of diverse receivers.

Although multi-rate multicast session is better in terms of “fairness", it is likely that single-rate sessions still exist due to application constraints, such as a requirement that all receivers must finish receiving all the data approximately at the same time. But for the multimedia application, which generally does not need this requirement, multi-rate transmission is definitely better than the single rate transmission in a way that the first one can cater the bandwidth variation from 56 kbps modem connection to a high-speed T1 link.

In this paper, we will exam the layered multicast, which is a multi-rate transmission for a multicast session. Layered multicast will be introduced in the next section.

1.3 Layered Multicast

Basically, the layered multicast is based on layered transmission scheme. In layered transmission scheme, data is distributed across a number of layers, which can be incrementally combined to provide progressive refinement.

The layered multicast was first suggested by Deering [1]. The idea of layered multicast is to encode the source data into a number of layers. Each layer is a separated multicast group, with receivers deciding to join/leave a group on their own based on the network condition.

Assume the data that is going to be transmitted can be distributed into *l* multicast groups with bandwidths L_i, i=0,…l-1. Receivers can adjust the transmission rates by using the cumulative layered transmission scheme.

$$B_i = \bigcup_{j=0}^{j=i} L_j$$

Now the adaptation to heterogeneous requirement becomes possible because it can be done independently in each receiver. Based on the network condition, a particular receiver can subscribe a bandwidth B_i by joining the L_0, L_1…L_i layers. The more layers the receiver joins, the better quality it gets. As a consequence of this approach, different receivers within a session can receive data in different rates. Also, the sender does not even need to take part in the congestion control.

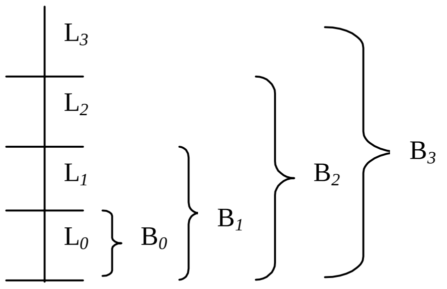

Figure 1

1.4 Fairness

Fairness has been a research topic in the Internet for many years and it is still one of the active research areas today. Basically, fairness issue in Internet is to study how the Internet allocates the bandwidth to each network session in a fair manner. Whether a new Internet traffic (like layered multicast) is proposed, it is definitely worthwhile to study fairness to avoid any unfair situation occurred in the Internet.

One of the well-accepted definitions of fairness is the max-min fairness [7]. Under a unicast environment, a bandwidth allocation is said to be max-min fairness if and only if each network session has a bottleneck link where the bandwidth allocated for that session is maximum among all the competing sessions with that link.

However, TCP is not max-min fair. The congestion control for TCP was first introduced by V. Jacobson [10] in 1988. Various papers [3], [6] have analyzed the theoretical throughput behavior of the TCP congestion control. For a low loss rate (< 0.01) and under normal operation conditions, the throughout of TCP at steady state is:

$$Throughput = F = \frac{C \times MTU}{RTT\sqrt{p}} \quad \text{--}(1)$$

where,

Throughput(*F*) is the data sent, measured in bytes/second.

C is a constant between 0.9 and 1.3.

MTU is the maximum transfer unit (i.e packet size, in bytes).

RTT is the round trip time in second.

p is packet loss rate, between 0 and 1.

Considering equation (1), it is noted that the TCP throughput is inversely proportion to RTT. In other words, if the TCP session has less round trip time, it will get more bandwidth allocations. Although TCP is not max-min fair, it does have another type of fairness (Less RTT, More Throughput) and we call it TCP fairness.

In multicast, the situation is much complicated. First of all there is no consensus on the fairness issue between multicast and unicast traffic. Should a multicast session be treated as a single session, which deserves no more bandwidth than a single TCP session when they share network resources? Or should the multicast session be given more bandwidth than TCP connection because it intends to serve more receivers? If the latter argument is true, how much more bandwidth should be given to the multicast session and how do we define fairness in this case?

In 1999, Dan Rubenstein [7] extended the definition of max-min fairness for unicast session to multicast session and proved that only multi-rate transmission can achieve the extended max-min fairness. It is consistent with what we have discussed in section 1.2 – multi-rate multicast is fairer than single-rate multicast. In theory, multi-rate multicast can be as max-min fair as the unicast.

However, to make the layered multicast max-min fair is not realistic. Most traffic in the Internet is TCP and TCP is not max-min fair. A network traffic, which is max-min fair with itself, does not mean that it is fair with TCP (Actually, it is unfair with TCP). To ensure the layered multicast and TCP can get the bandwidth allocation in a fair manner, layered multicast needs to be TCP fair. One of the key elements for TCP fairness is that the bandwidth allocation for the layered multicast needs to be inversely proportional to RTT.

Another thing that can affect the fairness is the layering in layered multicast. Consider a link with bandwidth 64 kbps consists of layered multicast and TCP traffic. In the layered multicast, the sender provides several layers and the transmission rate of each layer is 64 kbps. Then the layered multicast either occupies the whole link (64 kbps) or it cannot get any bandwidth allocation and the whole link is occupied by TCP.

In general, we can obtain more desirable fair bandwidth allocation for layered multicast if the number of layers can be finely divided. One way to obtain these layers is to have the sender configure layers so that each receiver can obtain its fair rate by joining some subset of the layers. However, the number of layers can be as large as the number of receivers in the session, making such an approach infeasible for large multicast sessions. Furthermore, it is too difficult for a sender to obtain the feedback needed to appropriately configure the rates of each of the layer. Also the number of layers and the rate per layer is often beyond the control of the sessions itself, due to application-specific requirement that limits the availability of multicast groups (layers).

In this paper, we will not discuss the layering in layered multicast. We just want to highlight another layering issue that can affect the fairness in layered multicast and suggest further research.

The TCP friendly congestion control for layered multicast is the main topic of this paper.

1.5 Objectives

The main objective of this project is to study the TCP fairness for layered multicast. We attempt to establish an effective TCP friendly congestion control algorithm for layered multicast. More specific, the congestion control should be:

- Fair with TCP – The congestion control should illustrate the following TCP fairness properties.
  - Throughput should be inversely proportion to RTT.
  - For any TCP and layered multicast with the same data path (same source, intermediate nodes and receiver),

    $T_{TCP} = T_{LM}$

    Where

    T_{TCP} - the long term average throughput for TCP.

    T_{LM} - the long term average throughput for Layered Multicast.

(Because of the limitation of the layering, it is impossible to achieve equivalent bandwidth allocation for TCP and layered multicast when receivers are restricted to join some arbitrarily chosen fixed set of layers. That why the long term average throughput is being used in the definition.)

- Inter-protocol fairness – Other than TCP, layered multicast should be fair with another layer multicast sessions.

- Efficiency – Each data path of each session should have a bottleneck link in a way that the bandwidth allocation would exceed the capability of that link if a particular receiver of that data path subscribes an additional layer provided that another session in that link does not decrease the bandwidth allocation. Basically, this property ensures that the bandwidth of the link is fully utilized.

- Fast convergence rate – The congestion control should converge to the steady state quickly.

- Avoid congestion collapse – For each link within a session, the bandwidth allocated to that session in that link should be equal to the maximum receiving rates among the receivers which the data paths of those receivers contain that link.

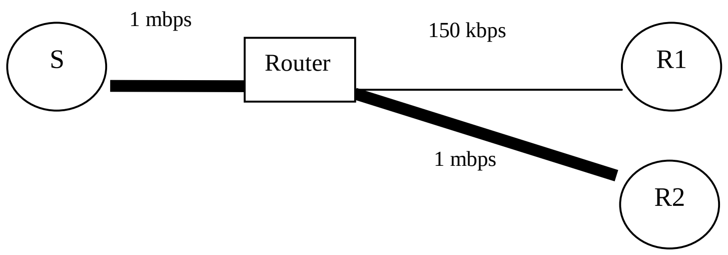

Figure 2

We illustrate the congestion collapse by using above diagram. Suppose S is the source of unresponsive multicast flow with transmission rate 1 mbps and R1 is the only receiver in the multicast group. Since the bandwidth of the router-R1 link is just

150 kbps, the maximum of the receiving rate in the multicast session is 150 kbps while the transmission rate is 1 mbps. Therefore, most of the packets will drop at router. This kind of congestion collapse was illustrated in [3]. Basically the bandwidth is wasted by delivering packets though the network that are dropped before reaching their ultimate destination.

Congestion collapse causes unfairness bandwidth allocation and wastes the network bandwidth. Suppose a TCP connection is established between S and R2. The TCP flow will reduce the sending rate in response to congestion, leaving the unresponding multicast flow to use the available bandwidth. Actually, the congestion collapse is primarily due to the unresponding flow which is without a proper end to end congestion control.

1.6 Solutions and Contributions

In our attempt to design and evaluate an effective “TCP friendly” congestion control algorithm for layered multicast.

- TCP fairness for layered multicast – We study TCP fairness properties for some of the existing congestion controls (Priority dropping, RLM and RLC) for layered multicast.

- Middle layer dropping (MLD) and Entire layer dropping (ELD) mechanisms – We design and evaluate the two new congestion controls mechanisms – MLD and ELD. We also show that MLD and ELD could work well and exhibit TCP-friendly behavior. Both MLD and ELD are an end to end congestion control with “router assistance”, which the router will drop packets of a particular layer when congestion occurs.

1.7 Dissertation Overview

This dissertation is organized as follows:

- Chapter 2 reviews related works and addresses the TCP fairness issues for some of the existing congestion controls for layered multicast.

- Chapter 3 describes the MLD and ELD. We also present how we come up with the MLD and ELD and show the simulation results that prove that MLD and ELD are TCP friendly.

- Chapter 4 provides further analysis for MLD and ELD. Other than TCP fairness, we take a look at the other aspects of MLD and ELD.

- Chapter 5 shows the summary table and compares among the existing congestion controls for layered multicast, MLD and ELD.

- Chapter 6 briefly discusses the applications that are suitable for layered multicast.

- Chapter 7 is the conclusion.

2. Previous Works

Layered multicast congestion control has been studied since 1993. Quite a number of proposals have already been published. In this chapter, we review some of the related works.

2.1 Priority Dropping

After the layered multicast suggested by Deering [1] published in 1993, priority dropping [2] was the first congestion control method proposed for the layered multicast. The idea of priority dropping is that packets belonging to the base layer can be marked as high priority while packets belonging to each successive enhancement layer can be marked as successively lower priority. During the time of congestion, the network can preferentially drop the low priority packets and protect the base layer from significant loss. To achieve this goal, a drop-preference packet discard policy is added to all the routers in the network.

In theory, priority dropping can divert loss away from the more important packets to less important packets so as to improve the overall multimedia quality. Figure 3 shows the quality between the priority dropping and uniform dropping against the network load.

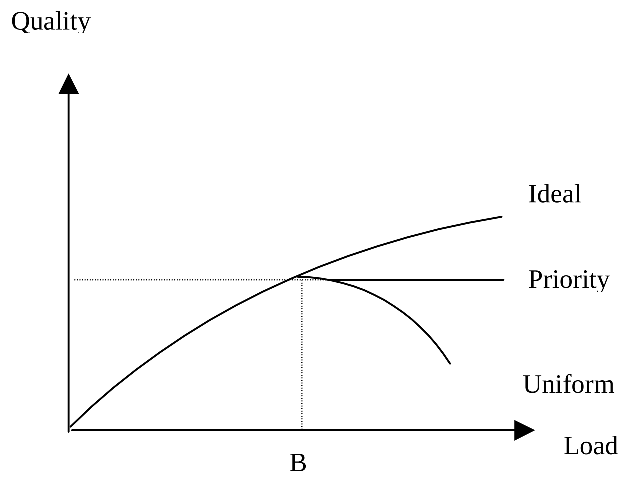

Figure 3

The uppermost curve is the ideal case if the network had infinite bandwidth. B is the bandwidth of the bottleneck link. Since the priority dropping will only discard less important packet, the quality can still be retained when congestion occurs - network load exceeds B. However in uniform dropping, quality decreases significantly beyond B because packets drop uniformly across all layers.

Although priority dropping can retain the quality for the layered multicast when congestion occurs, there are some disadvantages in priority dropping. First, priority dropping is not an end to end congestion control. Both source and receiver are not required to take part in reacting to congestion and adjusting their transmission rates accordingly. The layered multicast is similar to the unresponsive UDP flow, which is extreme TCP unfair. Second, without the end to end congestion control in priority dropping, it also causes congestion collapse [3].

2.2 Receiver-driven Layered Multicast (RLM)

Receiver-driven Layered Multicast (RLM) [4] is the first well-known end to end congestion control for layered multicast. In RLM, receiver detects network congestion when it observes increasing packet losses. Receiver reduces the level of subscription if it experiences congestion. In the absence of loss, the receiver estimates the available bandwidth by doing the so-called join experiments when the join-timer expires. A join experiment means that a receiver increases the level of subscription and measures the loss rate over a certain period. If the join-experiment causes congestion, the receiver quickly drops the offending layer. Otherwise, another join-timer will be generated randomly and the receiver retains the current level of subscription and continues to do the join experiments for the next layer once the newly generated join-timer has expired.

To avoid the transient congestion that impacts the quality of the delivered signals, RLM do the join-experiments infrequently when they are likely to fail. RLM implement this strategy by increasing the join-timer exponentially when congestion is detected and a layer is dropped.

However, join experiments can interfere with each other. For examples, a receiver can misinterpret the congestion caused by its join experiment and mistakenly reduce the level of subscription even the congestion is induced by another receiver in a join experiment. To avoid this problem, RLM introduce a “shared learning” mechanism in which a receiver notifies the entire group by multicasting a message identifying the experimental layer before conducting a join experiment. Receiver can only do the join experiment for the layers, which is equal to or below the newly multicast experimental layer.

In general, the subscription level can be increased or decreased in RLM based on the following rules:

1. Before doing the join experiment, receiver will perform the “shared learning” by broadcasting a notification message to all receivers in the multicast group. By doing so, all the receivers will know which layer is currently participating the join experiment.

2. Join-timers are randomized to avoid protocol synchronization effect. If a join-timer expires and no experiment or a lower layer experiment is in progress, receiver will perform the join experiment to increase the level of subscription. Otherwise, the current join-timer is ignored and a new one will be generated.

3. If a packet loss is detected, depending on different circumstances of the receiver, following actions will be taken:

    - If the receiver is currently participating the join experiment for the highest level, receiver will drop the offending layer and back off the join-timer;

    - If the receiver is currently doing join experiment but not for the highest level or no experiment is being performed, RLM will measure the long term congestion before dropping the offending layer.

RLM is the first end to end congestion control mechanism for layered multicast to avoid congestion collapse and can ensure the link is fully utilized. However, there are a number of problems in RLM.

<u>Convergence time:</u>

Although the “shared learning” can improve the overall scalability of RLM, sharing learning increases the convergence time to steady state especially for a large number of receivers and some receivers have spare capacity. Receiver with a larger bandwidth capacities need to wait for other receivers with lower bandwidth capacities to reach their steady state before it can join additional upper layer. Therefore, if the number of receivers grows, we expect longer convergence time since a large number of receivers will suppress the join experiment at higher layer.

TCP fairness:

Another challenge in RLM is that it is not fair with TCP. We consider a model with only one source and one receiver. Since the increasing rate of RLM depends on join-timer, NOT RTT, the bandwidth allocation for RLM should be constant for a long run under this model regardless of RTT. However, the increasing rate of TCP is inversely proportional to RTT. As a result, TCP would get more bandwidth for short RTT while it will get less for long RTT.

Intra-Protocol fairness:

Other than TCP, RLM is not even fair with itself. As mentioned in point (1), different network topologies will have different convergence times in RLM. Because of sharing learning, in general, the network models with more receivers will have slower increasing rate while those with fewer receivers will have faster increasing rate. We can anticipate that the network topologies with fewer receivers can get more bandwidth allocations than those with more receivers because of the relatively faster increasing rate. Therefore, suppose there are 2 RLM sessions with the same source, the bandwidth allocation of these 2 RLM sessions may not be the same in a particular receiver, which subscribes both RLM sessions.

2.3 Receiver-driven Layered Congestion Control (RLC)

RLM does not exhibit TCP-friendly behaviors. To address this problem, L. Viscisano, L. Rizzo and J. Crowcroft [5] proposed another receiver-driven layered congestion control mechanism (RLC), which solves some fairness issues. Similar to RLM, RLC uses packet drops signals to indicate the congestion at the receiver level. However, in RLC, sender drives the join experiment. Receivers can only attempt to increase the level of subscription immediately after each synchronization point (SP), which is a specially flagged packet sent by the sender at each layer. Basically the idea of SP is to impose some synchronization among the receivers. To overcome the problem that a receiver misinterpret the congestion and mistakenly reduce the level of subscription if the congestion is induced by another receiver in adding a new layer, SP at each layer is always a subset of the SP on the previous layer.

Since the leave delay of a multicast group is expensive, the subscription level can only be increased when it is certain that the attempt can be successful. To estimate the available bandwidth, sender regularly generates the short bursts of packets. During bursts, which have a during γ_o[1], two back-to-back packets are sent at each transmission. Following the bursts, there is an interval γ_o during, which the transmission is suspended. If no loss is experienced during bursts, the subscription level will be increased. SP is located at the end of each burst.

It takes some time to complete the leaving phase of a multicast group. A receiver may continuously experience packet losses during leaving phase and decrease the subscription multiple times in response to a single failed join. To overcome this problem, RLC introduces a deaf period t_D. A receiver does not react to further losses for a time t_D after a loss is detected and the subscription level has been decreased.

[1] γ_o is the inter-packet time at base layer. Actually $\gamma_o = \frac{MTU \times 8}{L_0}$

Basically, the congestion control mechanism is that:

1. Decrease if a loss is experienced.

2. Increase at SP if no loss is detected during the burst.

3. Unchanged if loss is detected only at burst or during the deaf period.

Although RLC improves some fairness issues, it is not fair with TCP. Also slow convergence is another issue in RLC. Another weakness of RLC is that burst induces losses, which probably reduce the user-perceiving quality.

<u>TCP fairness:</u>

Authors in [5] derive the approximation of the RLC throughput.

$$F = \frac{C' \times MTU}{\gamma_0 \sqrt{PW} \sqrt{p}} \text{ -- } (2)$$

where,

F is the throughput, measured in bytes/second.

C' is a constant between 1 and 2.

MTU is the maximum transfer unit (i.e. packet size, in bytes).

W is the number of packets between two bursts.

γ_0 is the inter-packet time at the base layer.

P is the number of bursts between SP at the base layer.

p is packet loss rate, between 0 and 1.

Comparing this equation (2) with TCP throughput equation (1), they do exhibit some similarity. However it is not sufficient to say that RLC is fair with TCP. In fact, by looking at these two equations very carefully, we notice that TCP throughput is inversely proportional to RTT while RLM throughput is inversely proportional to $\gamma_o\sqrt{}PW$. In a multicast session, different receivers basically have different RTT but $\gamma_o\sqrt{}PW$ should be the same among all receivers. Unless the RTT of all the receivers are comparable with $\gamma_o\sqrt{}PW$, it can be predicted that RLC is not fair with TCP under the same sender and receiver.

Convergence time:

Another weakness of RLM is the slow convergence time because SP at each layer is always a subset of the SP at the previous layer. That means the upper layers will get less chance to attempt to join the next layers. If the number of layers is large, RLC will take a long time to converge to steady state.

Intra-Protocol fairness:

The increasing rate of RLC depends on the SP interval ($\gamma_o\sqrt{}PW$). Suppose two RLC sessions with the same SP interval, they should get the same share of bandwidth. If they have different SP interval, the one that has a shorter SP interval should get more bandwidth allocations. The reason is that a shorter SP interval means more frequent join experiment. Therefore, it has a faster increasing rate.

3. Middle Layer Dropping & Entire Layer Dropping

As discussed in the previous chapter, the bandwidth allocation for a particular receiver in RLM or RLC session is independent of RTT. However, in TCP, connections with shorter RTT get more bandwidth than those with longer RTT do. If our objective is to make the layered multicast congestion control TCP-friendly, we should consider introducing the RTT factor in the layered multicast congestion control. Furthermore, our design should also consider the Additive Increase Multiplicative Decrease (AIMD) property of TCP.

Based on our observations, we introduce two new congestion controls in this chapter.

- Middle Layer Dropping (MLD)
- Entire Layer Dropping (ELD)

Both MLD and ELD are end to end congestion controls with router assistance which routers will selectively drop packets of a particular layer if congestion occurs. By simulation, we can prove that both MLD and ELD can achieve the TCP fairness.

3.1 Simulation Topologies

Before introducing MLD and ELD, we present the network topologies that are used for the simulation. NS is being used as the simulation tool in this project.

Two topologies are illustrated in figure 4 and 5. Figure 4 consists of a single source and a receiver separated by a bottleneck link. Figure 5 extends the first topology with multiple receivers with different RTT. S is the sender and R1…. R100 are the receivers of the multicast group. A and B are the drop-tail routers running DVMRP. The link between A and B is always the bottleneck link with bandwidth 150 kbps. All other links have higher bandwidth (i.e. 1 mbps). S sends a layer multicast with the bandwidth of each layer 24 kbps to all receivers R1…R100. To evaluate the TCP-fairness, S also establishes a TCP connection with R1.

1. One sender and one receiver – The RTT of link A-B is 400ms. The RTT of link S-A and link B-R are 200 ms.

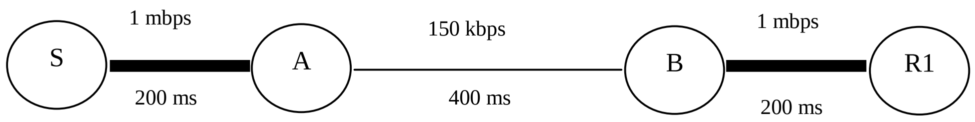

Figure 4

2. One sender and 100 receivers – In order to understand the impact of different RTT to the congestion control. Figure 5 is the case with one sender and 100 receivers. The range of RTT for links B-R1 to B-R100 is between 200ms to 2200ms.

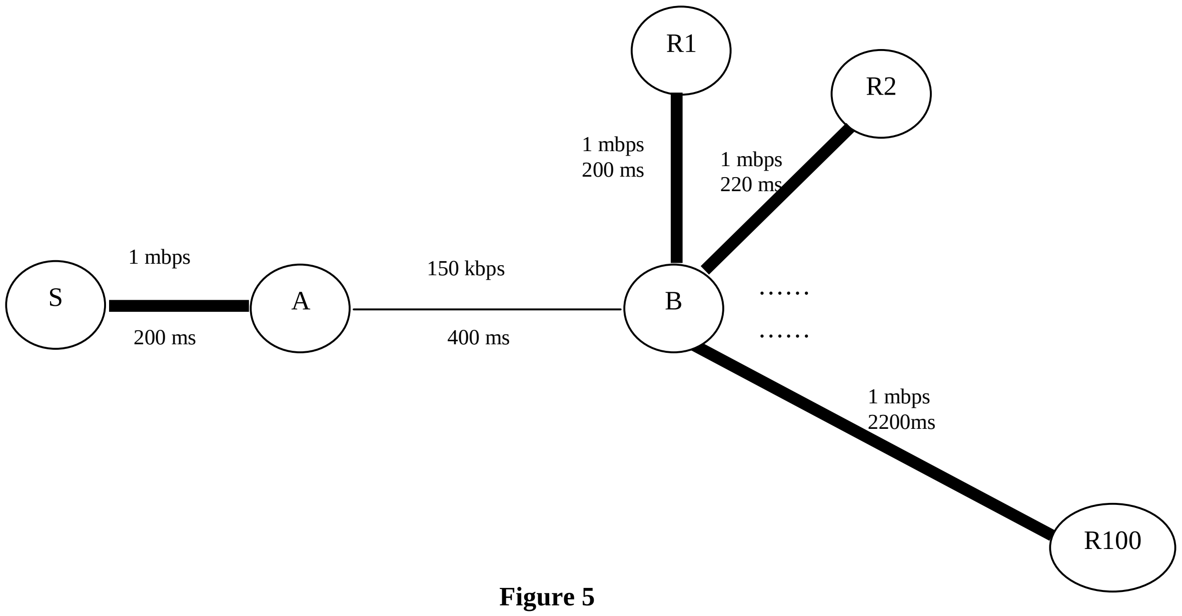

Figure 5

3.2 Additive Increase Multiplicative Decrease

To make the layered multicast "TCP like", we simulates the Additive Increase Multiplicative Decrease (AIMD) property in TCP for layered multicast. We establish the congestion control for layered multicast with the following properties:

1. In the absence of packet loss, receivers increase the level of subscription linearly in a step like fashion. But how long does a receiver need to wait before it attempts to add the next layer?

   To determine the join-timer, we simulate the behavior of TCP congestion window. In TCP, the congestion window is increased by roughly one packet per window for each RTT in the congestion avoidance phase. How long is a TCP sender required to increase the bandwidth allocation for L_{i+1} in the congestion avoidance phase? If we can determine this duration and use it as the join-timer for layered multicast, we basically can make the layered multicast and TCP with the same increasing rate as shown in figure 6. Also, the bandwidth allocation for them should be roughly the same.

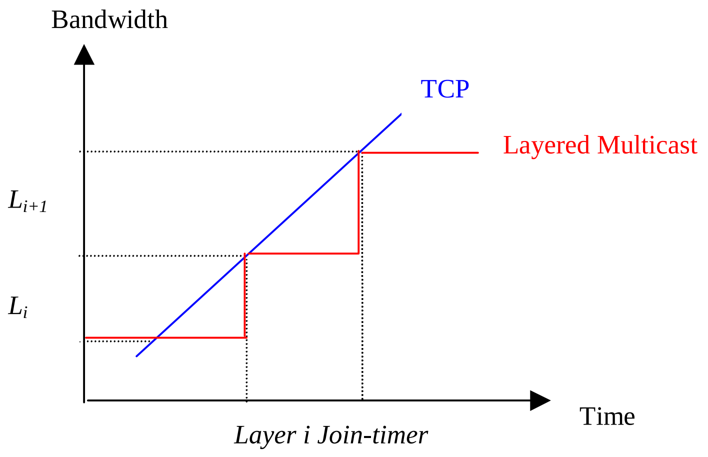

Figure 6

We use the following equations to estimate the join-timer of each layer for layered multicast. Suppose *cwnd* is the congestion window in TCP, then the average throughput for TCP connection during a RTT is:

$$Throughput = \frac{cwnd}{RTT} \times MTU \text{ --(3)}$$

Since the bandwidth allocation for a TCP session is increased by L_{i+1}, by equation (3), we can estimate the equivalent number of TCP congestion window increased.

$$L_{i+1} = \frac{cwnd}{RTT} \times MTU$$

Therefore,

$$cwnd = \frac{L_{i+1}}{MTU} \times RTT \text{ ---(4)}$$

Equation (4) shows the relation between congestion windows *cwnd* and L_{i+1}.

Since the duration that needs to increase one congestion windows is approximately equal to one RTT in TCP, based on equation (4), we can determine how long a TCP sender is required to increase the bandwidth allocation for L_{i+1} in the congestion avoidance phase.

$$cwnd \times RTT = \frac{L_{i+1}}{MTU} \times RTT^2 \text{ --(5)}$$

We use the same result as the join timer of each layer for layered multicast.

$$Layer\ i\ join\text{-}timer = \frac{L_{i+1}}{MTU} \times RTT^2 \text{ --(6)}$$

2. If packet loss is detected in TCP, the congestion window will be halved. To simulate this behavior in layered multicast, we consider just decreasing the subscription level to previous layer rather than reducing the subscription level to a particular layer that accumulates half of the bandwidth. By using equation (6), we

calculate the join-timer by assuming that the bandwidth has been reduced to one-half.

At layer i, the accumulative bandwidth is B_i. Suppose a receiver experiences packet loss at layer i, it reduces the subscription level to layer i–1. To calculate the join-timer, we assume that the bandwidth had been reduced to one-half. i.e. $B_i/2$. Based on equation (6), the join-timer for that particular receiver should be:

$$\frac{B_i}{2MTU} \times RTT^2$$

after reducing the level of subscription.

3. Because the leave phase takes a long time to complete. In order to avoid the multiple decreasing of the subscription level in response to a single congestion instance. After loss and a subsequent decrease in the subscription level, a receiver does not react to further loss for about 2 RTT.

In general, the congestion control will be as follows. Consider a particular receiver at layer i, it will:

1. Increase to i+1 after a period $\frac{L_{i+1}}{MTU} \times RTT^2$ if no packet loss is experienced.
2. Decrease to i-1 if a packet loss is detected. Increase back to i after a period $\frac{B_i}{2MTU} \times RTT^2$ if no loss is experienced.

3. After subscription level has been decreased, a receiver will not react to further losses for about 2 RTT.

3.3 Simulation – AIMD

We evaluated this model through simulation topologies described in section 3.1.

Topology 1 - One sender and one receiver: By running the simulation 3600s, we got the results shown in the following table:

	TCP	**Layered Multicast**
Bandwidth allocation	70.39 kbps	63.01 kbps

Because of the TCP-like AIMD congestion control for layered multicast, both layered multicast and TCP can get roughly the same share of bandwidth for a single source and receiver.

Layered multicast gets a little less bandwidth than TCP does. The primary reason is that layered multicast has a limited set of throughputs allowed while TCP is able to adjust its throughput with much granularity. In AIMD congestion control, the receiver will not attempt to join the next layer until the join-timer expires. Therefore during join-timer, the bandwidth allocated for layered multicast is constant but bandwidth allocated for TCP is increasing.

Topology 2 - One sender and 100 receivers: By running the simulation 3600s again, we got the average throughput for TCP is 80.66 kbps. Average bandwidth allocation for layered multicast for each receiver with different RTT is shown in the following graph:

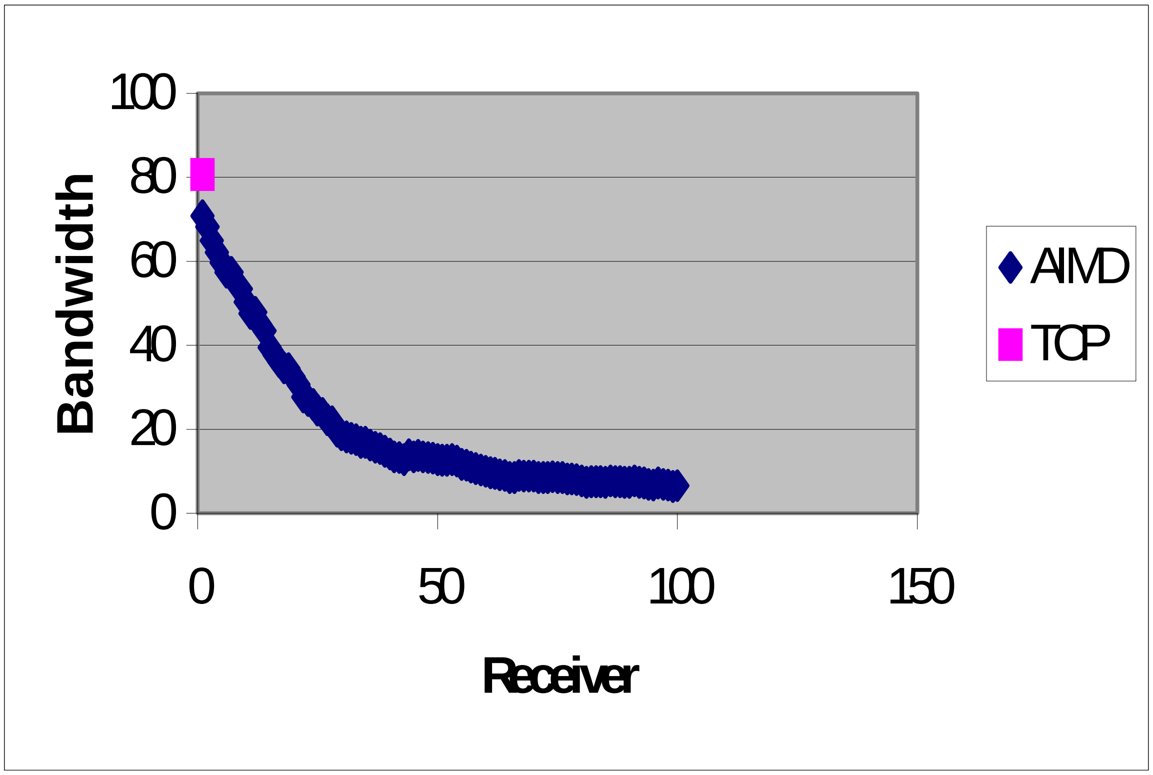

Figure 7

We note that the bandwidth allocation for both layered multicast and TCP are roughly the same at receiver R1. We also note that, in layered multicast, the bandwidth allocation for other receivers decrease when their RTT increases. Therefore, according to our definition, AIMD congestion control is TCP-friendly. However, there is some tradeoff between TCP fairness and intra-fairness. In this model, receivers with long RTT cannot get any bandwidth allocation.

The main reason is that each receiver will decrease the subscription level when packet loss is detected regardless of whether the loss is generated by itself. Receivers with long RTT cannot be allocated sufficient bandwidth because their increasing rates are fewer than those with shorter RTT and the packet loss generated by other receivers with shorter RTT will also cause them to dropping the subscription level.

It is the reason why RLM and RLC need to introduce some synchronization (Share Learning and SP) among the receivers to prevent a receiver from misinterpreting the congestion and mistakenly reduces the level of subscription if another receiver induces congestion.

3.4 AIMD with Priority dropping

To overcome the above problem and improve the intra-fairness, we consider the priority dropping mechanism, which has already been discussed in section 2.1, together with AIMD end to end congestion control. Basically, the idea of priority dropping is that when a packet of a particular layer i arrives and the queue is full, rather than dropping the arriving packet, the arriving packet is queued and another packet of the highest layer j in the queue is dropped at the congested router. By doing so, the packet loss happened at the highest layer does not affect the receiver with a lower layer. Therefore, in theory, receivers, which are not at the highest level, cannot detect any packet loss and mistakenly reduce the level of subscription.

In general, the congestion control will be as follows:

1. At router level - A drop-preference packet discard policy is added to all the routers in the network.

2. At receiver level - The congestion control is the same as the AIMD congestion control.

However, we will illustrate in the next section that priority dropping mechanism together with AIMD end to end congestion control is not TCP-friendly.

3.5 Simulation - AIMD with Priority Dropping

The simulation results are shown as below:

Topology 1 - One sender and one receiver: By running the simulation 3600s, we got the results shown in the following table:

	TCP	**Layered Multicast**
Bandwidth allocation	62.46 kbps	72.62 kbps

Compared with those results without priority dropping, layered multicast gets a little (9kbps) more bandwidth allocations in this case. During congestion and after subscription level has decreased, receivers could not detect any further packet loss even the leave phase has not completed because, in priority dropping, only packet of the highest layer in the congested router will drop. Therefore, receivers will not react to the further packet loss in response to a single congestion instance. It is the reason why AIMD with priority dropping can get a little more bandwidth than AIMD without priority dropping.

Other than that, we note that the bandwidth allocation between layered multicast and TCP are still roughly the same. Basically under the circumstance of only one source and receiver, there should not be any major difference between AIMD congestion control for layered multicast with or without priority dropping.

Topology 2 - One sender and 100 receivers: We run another simulation for 3600s for a single source with 100 receivers. The results are shown as below:

1. Average TCP throughput = 19.14 kbps. TCP gets much less bandwidth than layered multicast.

2. Average bandwidth allocation for layered multicast for each receiver with different RTT is shown in the following graph:

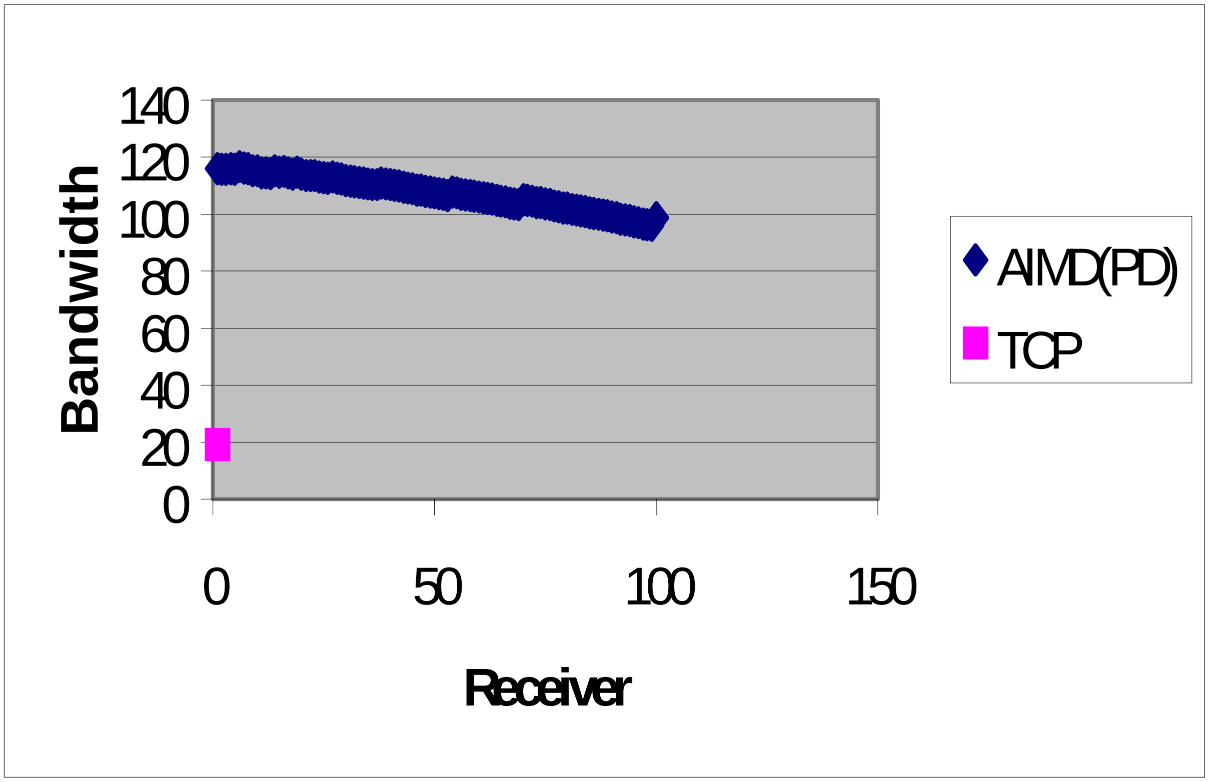

Figure 8

The bandwidth allocation between layered multicast and TCP is not in a fair manner. Layered multicast is more aggressive than TCP. The reason is that when packet loss occurs in priority dropping mechanism, receivers with a highest subscription level will drop the highest layer and TCP will halve its congestion windows. However receivers at the lower level cannot detect any packet loss and they have no idea of when the congestion occurs at the highest level. Without any congestion signal, they will continuously increase the subscription level to the next layer without any delay. As a result, the subscription rate will increase if the number of receivers increases. That why the TCP is nearly shutdown when there are too many receivers in a layered multicast session.

3.6 Middle layer dropping mechanism (MLD)

In AIMD congestion control without priority dropping, receivers with long RTT cannot get sufficient bandwidth. To tackle this problem, we use priority dropping together with AIMD congestion control. However, AIMD with priority dropping is not TCP friendly. When the number of receivers increases, layered multicast gets more bandwidth than TCP does. The main reason is that receivers, which are not at the highest level, do not know when the congestion occurs.

Is there any way that we can let other receivers, which are not at the highest layer, know any congestion occurred? By doing that, those receivers, which are not at the highest layer, can prolong their join-timer so as to reduce their increasing rates to avoid that layered multicast is too aggressive than TCP.

To achieve this goal, we modify the drop-preference packet discard policy in all the routers for priority dropping. Instead of dropping the packet of highest layer, we consider dropping the packet of the "middle" layer when congestion occurs. The "middle" layer means a specific layer, which contains the median of the total bandwidth subscripted by the receivers. For example, suppose the bandwidth of each layer is 24 kbps in a layered multicast and if the highest layer is 3, the middle layer will be 2. If highest layer is 4, the middle layer will be 2. If highest layer is 5, the middle layer will be 3…

When congestion occurs, packet will be dropped at the “middle layer”. The packet loss is a signal to indicate any congestion occurred in all receivers with more than half of the total bandwidth subscribed. Therefore, receivers can adjust their increasing rates or reduce the level of subscription based on the packet loss signal.

For each receiver, the congestion control mechanism is almost the same as AIMD with priority dropping except that each receiver will determine whether the lost is generated by itself. For example, if the subscription level of a particular receiver is 2 and it detects a packet loss at layer 2. It will not decrease its subscription level because the packet loss should be caused by layer 3 or 4. Figure 9 shows the reaction of each receiver during congestion:

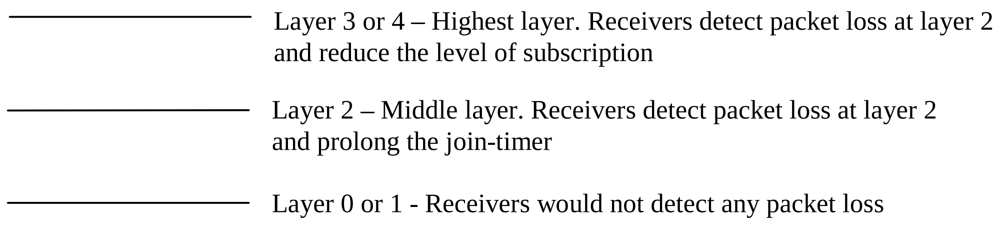
Layer 3 or 4 – Highest layer. Receivers detect packet loss at layer 2 and reduce the level of subscription
Layer 2 – Middle layer. Receivers detect packet loss at layer 2 and prolong the join-timer
Layer 0 or 1 - Receivers would not detect any packet loss

Figure 9

In general, if packet loss is detected, receivers, which are not at the highest layer, will ignore the current join-timer and schedule a new one by the following equation.

$$\text{Join-timer} = \frac{G}{2MTU} \times RTT^{2}$$

where G is the total bandwidth accumulated from layer i to layer k. Layer k is the current layer subscription for the receivers and i is the middle layer. Basically, we use this equation to assume that the bandwidth of each receiver is reduced to middle layer during congestion.

3.7 Simulation – MLD

Topology 1 - One sender and one receiver:

	TCP	**Layered Multicast**
Bandwidth allocation	65.88 kpbs	72.99 kpbs

Topology 2 - One sender and 100 receivers:

1. Average TCP throughput = 60.66 kbps.

2. Average bandwidth allocation for layered multicast of each receiver with different RTT is shown in the following graph:

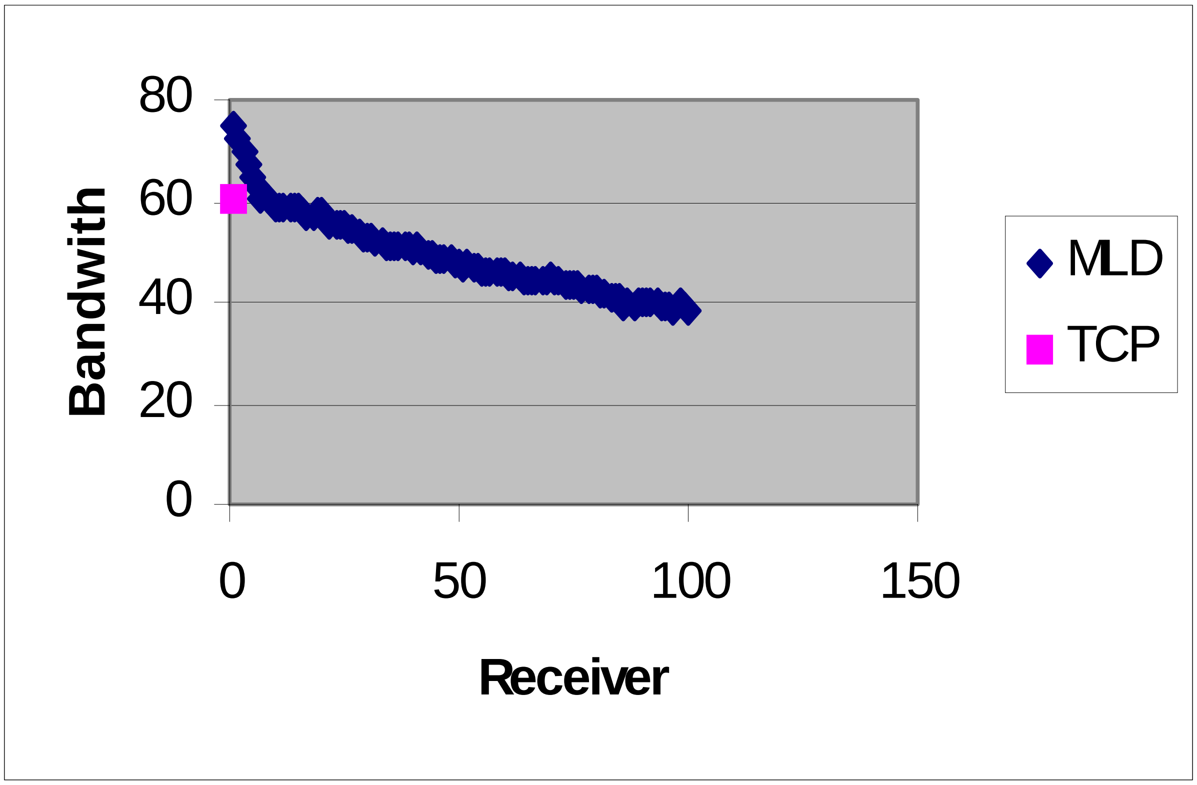

Figure 10

According to figure 10, both TCP and layered multicast roughly get the same share of bandwidth. Even receivers with longer RTT can get at least half of the maximum bandwidth allocated for layered multicast. In MLD, routers will selectively drop the middle layer when congestion occurs. All the receivers can have a signal to indicate any congestion occurred when they subscribe more than half of the total bandwidth. Therefore MLD can prevent layered multicast from becoming too aggressive and also it can make sure that receivers with long RTT can get at least half of the total bandwidth of the layered multicast. Basically, MLD improves the intra-fairness while preserving the TCP-fairness.

3.8 Entire layer dropping mechanism (ELD)

The implementation of MLD is problematic in real situation. The difficulty is that router does not have an idea of what the middle level should be because the transmission rate of each layer can be arbitrary. For example, if the transmission rate of each layer is exponential increasing (2kbps, 4kbps, 8kbps…), the middle layer is the highest subscribed layer. On the other hand, if the transmission rate is constant in each layer, the middle layer should be the highest subscribed layer divided by 2. The problem is that router does not maintain the transmission rate during each connection. Unless we can limit a particular set of layering (like exponential or constant) or implement any additional mechanism, router cannot calculate the middle layer at any particular time when congestion occurs.

Because of this reason, we develop another congestion control mechanism – entire layer dropping mechanism (ELD), which basically is a modified version of MLD. And it should be easier to implement ELD than MLD.

In ELD, when a packet of a particular layer i arrives and the queue is full, the arriving packet will be discarded and the highest layer j will be totally dropped at the congested router. When the receiver detects any packet loss, indication of congestion will be given out and the receiver will then adjust the increasing rate by reset the join timer. The congestion control will be as follows:

1. At router level – To completely drop the highest layer during the congestion, each router in the network is configured to remove the routing path of the highest layer if the queue is full. One of the great differences between ELD and MLD is that it is the router now responsible for reducing the level of subscription rather than the receiver. After the routing path of the highest layer is dropped by the congested router, the subscription for that particular highest layer will eventually expire at the receiver end.

2. At receiver level – If there is no packet loss detected. Increase to next layer, i+1, after a period $\frac{L_{i+1}}{MTU} \times RTT^2$ (Same as the AIMD congestion control). If packet loss is detected at receivers, receivers will then reset the join-timer by following the same MLD equation. Suppose the current subscription is k. If packet lost is detected at layer i, the new join-timer is $\frac{G}{MTU} \times RTT^2$, where G is the total bandwidth accumulated from layer i to layer k.

3. Unlike MLD, the packet drop is randomized in ELD. However, since the median of the layers for all packet drops should be the "middle" layer, we should expect the same result with MLD.

3.9 Simulation – ELD

Topology 1 - One sender and one receiver:

	TCP	**Layered Multicast**
Bandwidth allocation	75.88 kbps	60.18 kbps

Topology 2 - One sender and 100 receivers:

1. Average TCP throughput = 60.66 kbps

2. Average bandwidth allocation for layered multicast for each receiver with different RTT is shown as following graph:

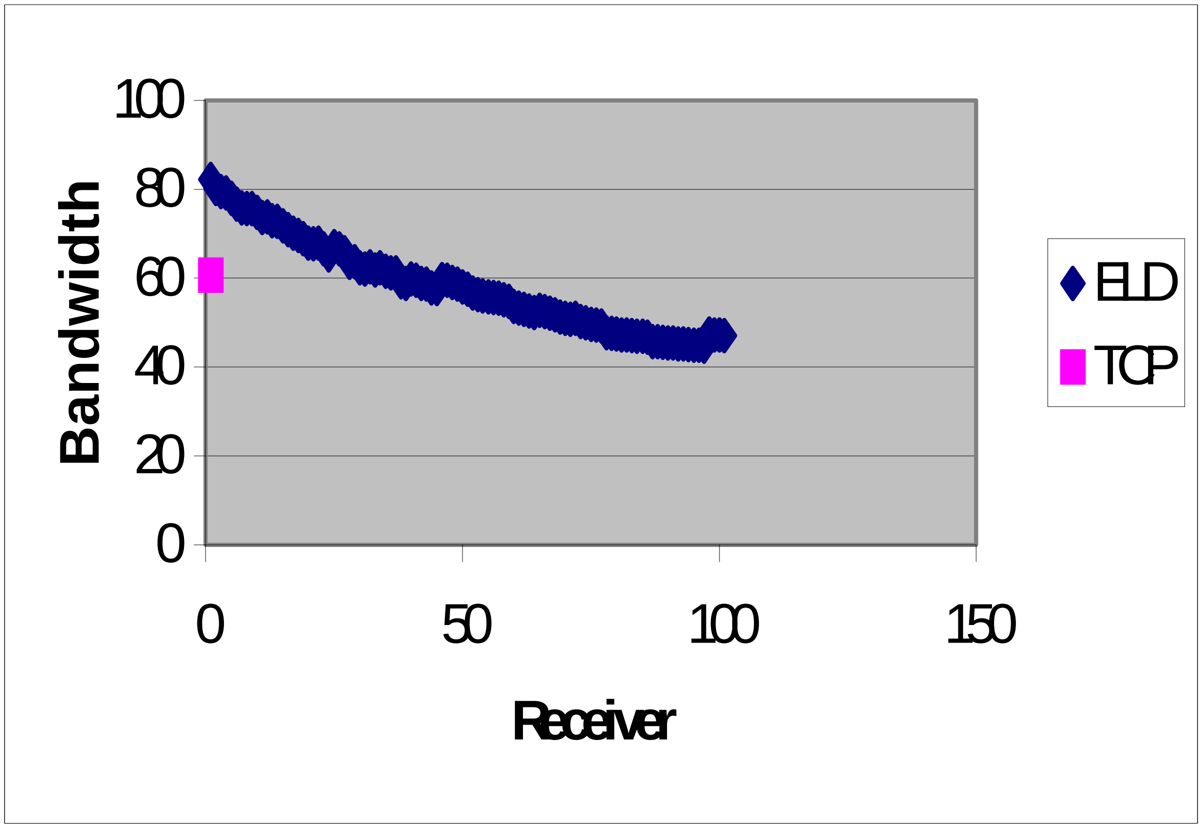

Figure 11

The results is similar to what we get from MLD. Receivers with short RTT get roughly the same bandwidth with TCP while receivers with long RTT can get at least half of the total bandwidth.

4. Further Analysis for MLD and ELD

In the previous chapter, we establish middle layer dropping (MLD) and entire layer dropping (ELD) mechanisms and prove that both mechanisms are TCP-friendly. In this chapter, we highlight other important aspects of MLD and ELD:

- Performance
- Avoid Congestion Collapse
- Inter-protocol fairness
- Convergence Time

4.1 Performance

One of the fundamental goals of the end to end congestion control is to ensure that the link is fully utilized while adjusting the transmission rate to avoid congestion. To illustrate the efficiency of MLD and ELD, we evaluate these two mechanisms through the second simulation topology described in section 3.1. We did two separated simulations for MLD and ELD and each simulation was run for 3600s. Source (S) sent a layer multicast with bandwidth of each layer 24 kbps to all receivers R1…R100. Figure 12 shows the simulation results. It shows the throughputs against the receivers with different RTTs for both MLD and ELD.

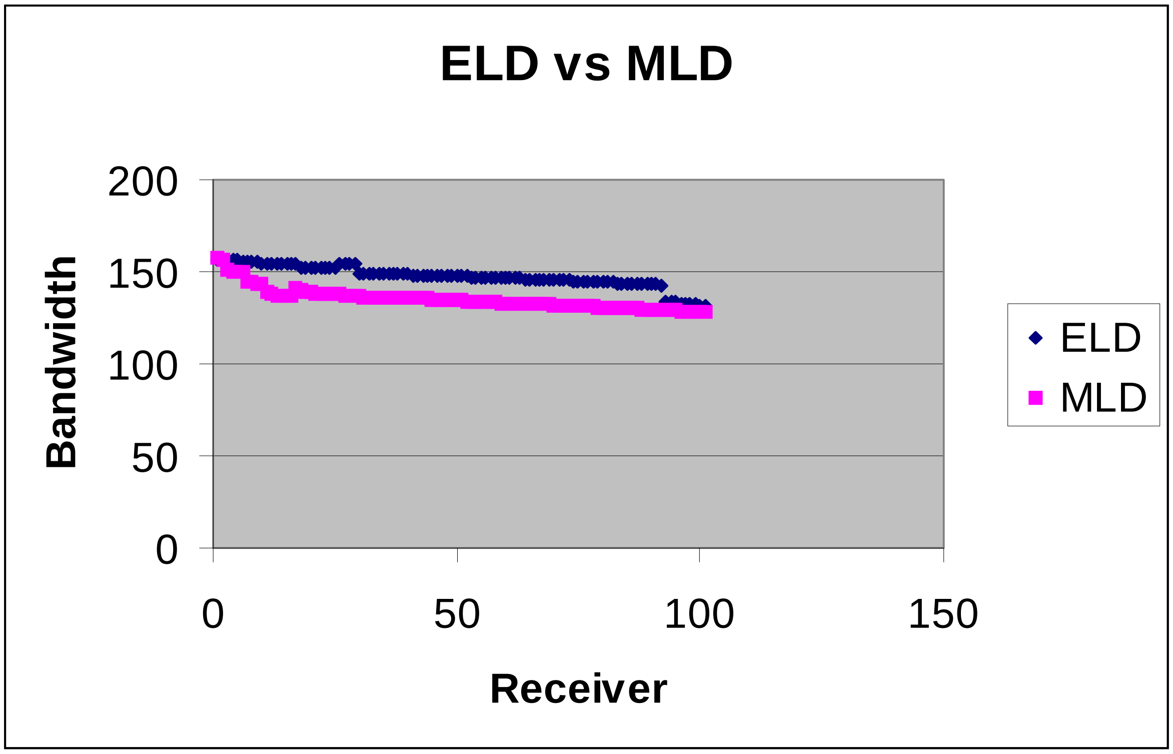

Figure 12

Based on the results, we note that:

1. Since the increasing rate is inversely proportional to RTT, receivers with shorter RTT can have a better performance than those with longer RTT. This is a tradeoff between TCP-friendly and performance. To make the MLD and ELD "TCP like", we introduce the RTT factor in the congestion control and the RTT factor induces the performance issues in MLD and ELD. Actually, this is exactly the same problem as the intra fairness issue that we mentioned in the previous chapter. Router assistance in MLD and ELD does make some improvements for the receiver with long RTT. As we can see in the Figure 12, receivers with long RTT still can get a sufficient amount of bandwidth allocation. Without router assistance, I believe the situation would be much worse.

2. ELD is more effective than MLD. In MLD, there is a chance that both highest and second highest layers are dropped during congestion. For example, if the middle layer is 2, the highest layer can be 3 or 4. During congestion, receivers at either layer 3 or layer 4 need to reduce the level of subscription. In MLD, only highest layer is dropped during congestion.

4.2 Avoid Congestion Collapse

It is obvious that both MLD and ELD can avoid congestion collapse. Whenever congestion occurs, systems that are running MLD or ELD can adopt the transmission rate to avoid congestion. Therefore the congestion collapse should not happen under MLD and ELD.

4.3 Inter-Protocol Fairness

Suppose two or more MLD or ELD sessions are running together, How should be the bandwidth allocation? Will the bandwidth allocation be in a fair manner? To evaluate the inter-protocol fairness, we consider a simple model with only one source and two receivers. Also we focus on MLD first.

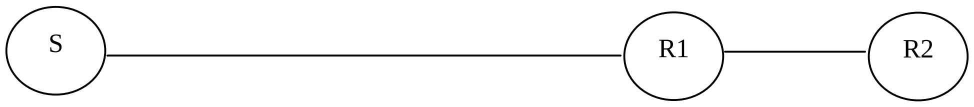

Figure 13

Figure 13 shows the network model. We establish two layered multicast sessions running MLD. Each MLD session has only one source (S) and one receiver (R1 or R2). Suppose i and j are the numbers of layers that R1 and R2 subscribed at the steady state respectively. Before the congestion, when the join-timer of receiver R1 expires, R1 increases the level of subscription by joining the next layer i+1. After a certain period, the join-timer of R2 expires and R2 subscribes the next layer j+1. Then the network is saturated and the router drops the packet at the middle layer. At the receivers, both R1 and R2 detect the packet loss at the middle layer and reduce the level of subscription back to i and j. Also R1 and R2 would reset their join-timers as below:

Joiner-timer of R1 - $\frac{B_{i+1}}{2MTU} \times RTT1^2$

Joiner-timer of R2 - $\frac{B_{j+1}}{2MTU} \times RTT2^2$

where *RTT1* and *RTT2* are the round trip times of link S-R1 and link S-R2.

B_{i+1} is the cumulative bandwidth at layer i+1.

MTU is the maximum transfer unit.

Without loss of generality, we assume that R1 will join the next layer i+1 before R2,

$\therefore$ Joiner timer of R1 ≤ Joiner timer of R2

$$\therefore \frac{B_{i+1}}{2MTU} \times RTT1^2 \le \frac{B_{j+1}}{2MTU} \times RTT2^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(7)$$

After R1 increased the level of subscription to layer i+1, the join-timer of R2 will then be expired before R1 attempts to do another subscription.

$$\therefore \frac{B_{i+1}}{2MTU} \times RTT1^2 + \frac{L_{i+2}}{2MTU} \times RTT1^2 \ge \frac{B_{j+1}}{2MTU} \times RTT2^2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(8)$$

where L_{i+2} is the bandwidth of layer i+2.

$\frac{L_{i+2}}{2MTU} \times RTT1^2$ is the join-timer of R1 at layer i+1.

By combing the equations (7) and (8), we get

$$\frac{B_{i+1}}{2MTU} \times RTT1^2 + \frac{L_{i+2}}{2MTU} \times RTT1^2 \ge \frac{B_{j+1}}{2MTU} \times RTT2^2 \ge \frac{B_{i+1}}{2MTU} \times RTT1^2 \ldots..(9)$$

By rewriting equation (9), we get

$$1 + \frac{L_{i+2}}{B_{i+1}} \geq \frac{B_{j+1} \times RTT2^2}{B_{i+1} \times RTT1^2} \geq 1 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots .(10)$$

Assume the bandwidth of each layer is constant and the subscription level is high. Then we can estimate that

$$\frac{L_{i+2}}{B_{i+1}} \approx 0$$

As a result,

$$\frac{B_{j+1} \times RTT2^2}{B_{i+1} \times RTT1^2} \approx 1 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (11)$$

When considering equation (11), we can conclude that if the *RTT2* is long, B_{j+1} will be small while if the *RTT1* is short, B_{i+1} will be large. Therefore, under several competing MLD sessions, receivers with shorter RTT can get more bandwidth allocations. It is similar to the property of the TCP fairness.

In ELD, it is difficult to devise the steady state analysis because the packet is randomly dropped rather than dropping the packet at the middle layer during congestion. We believe that ELD has more fluctuation than MLD at steady state because the possibility factor is introduced to generate the join-timer. The situation is much more complicated if there are many receivers in the multicast group. In general, we believe that the rule - less RTT more bandwidth, is valid for both MLD and ELD. However, further study for inter-protocol fairness is needed.

4.4 Convergence Time

In MLD or ELD session, receivers with relatively long RTT have more chance to detect packet loss and remain in the same layer before the join-timer expires. Therefore, if the multicast group is large, receivers with relatively long RTT will take more time to reach the steady state. As a result, we can expect that a multicast group has slow convergence time if it has large number of receivers with wide RTT range.

5. Implementation & Deployment

As mentioned before, MLD is difficult to implement in real situation because the router generally does not have an idea of what the middle level should be. In this section, we focus on some of the implementation and deployment issues of ELD. However, even ELD, there are quite a number of issues that need to be solved before we can deploy it in the Internet.

5.1 Router Assistance

ELD is an end to end congestion control with router assistance for layered multicast. Whenever congestion occurs, routers need to be able to stop routing and remove the routing path of the highest layer among those layers routed in the router. ELD requires that all the routers in the network have to possess this capability. To put this function into operation for all the routers in the Internet in a short period is not feasible. As a starting point, we can implement ELD in the Intranet first and gradually deploy it to the Internet.

Another issue in ELD is that both receivers and routers need to know the addresses of the layered multicast group. At this moment, the addressing for the layered multicast has not been defined yet in the Internet. This is another research area for layered multicast. One of the proposals is to put the addresses of the layered multicast sequential so that receivers and routers can get the addresses of other layers if they know the address of the base layer (Layer 0).

5.2 Parameters for the AIMD equation

Let us recall the AIMD equation.

$$\frac{L_{i+1}}{MTU} \times RTT^{2}$$

Therefore, in ELD, there are three basic elements that need to be determined – *MTU*, *RTT* and L_i.

MTU:

The MTU value can be determined by using the MTU discovery algorithm.

RTT:

Determine the RTT in a large multicast environment is problematic. All receivers send the ICMP echo requests will cause the feedback implosion problem. Basically, we have 4 different ways to determine RTT for a multicast group:

1. By using RTP, synchronize the time clock among all the receivers as well as source. Determine RTT using twice the delay from source to receiver. However the drawback of this method is inaccurate results in an asymmetric link.

2. Source sends a multicast signal with a number between 0 to 1 to all receivers. The number is the probability that the receivers are allowed to determine the RTT with source. To avoid feedback implosion problem, source then determines the probability number based on the number of RTT requests received. However the drawback of this method is not scalable well. When the size of the receivers increases, each receiver get less chance to determine the RTT and it means that the time interval for each RTT determination become longer.

3. Combine the method 1 and 2. Basically, we use method 2 to determine the RTT and adjust the RTT based on the value got from method 1. We believe it would improve the accuracy of RTT determination. However it still cannot completely solve the problem.

4. Implement an additional mechanism in router. Each router determines the RTT of its adjacent routers by regularly sending the ICMP echo. Define a special packet (e.g. new IGMP type) that the router will add the RTT value to this packet when it route via the router. In order to get the RTT, source will send this special packet to all receivers at based layer. However, the drawback is that it generates an additional traffic in the network and also it requires additional mechanism in the routers.

Further analysis is required to determine the best way to estimate the RTT. For all the simulations that we have done, we use method 1 to determine the RTT.

$\underline{L_i}$

How the receivers know the bandwidth of each layer is another problem that we need to solve. For all the simulations that we have gone through so far, we assume that receivers know this information. However, it may not be the case in real situation. Receivers need to determine the bandwidth of each layer in order to calculate the join-timer. By determining the no. of packets received over a certain period of time and low-pass filter, receivers can estimate these figures.

$$L_i = \alpha L_i + (1-\alpha)M$$

where M is the most recent measurement for the no. of packets received multiply by MTU over the period of measurement. α is the smoothing factor and L_i is regularly updated for each period of measurement (i.e. 1 second).

6. Summary Table

The following table is the comparison between priority dropping, RLM, RLC, MLD and ELD

Protocols	*TCP-fairness*	*Efficiency*	*Intra-protocol fairness*	*Avoid Congestion Collapse*	*Convergence Rate*
Priority Dropping	No	Yes	No	No	Fast. No steady point for priority dropping
RLM	No	Yes	Depends on the number of receivers	Yes	Slow when the receiver group is large
RLC	No	Yes	Depends on period of SP	Yes	Slow when the number of layers is large
MLD	Yes	Receivers with short RTT have a better performance than those with long RTT	Depends on RTT	Yes	Slow when the RTT range of the receiver group is wide
ELD	Yes	Same as MLD	Depends on RTT	Yes	Same as MLD

The following table shows which network components (source/routers/receivers) are involved in congestion controls.

Protocols	***Source***	***Router***	***Receiver***
Priority Dropping	No	Yes	No
RLM	No	No	Yes
RLC	Yes	No	Yes
MLD	No	Yes	Yes
ELD	No	Yes	Yes

7. Applications

The MLD and ELD congestion controls that we present here are for any kind of layered multicast. Multimedia application (like audio/video application, multi-media conferencing…) is one of the potential applications for layered multicast using MLD or ELD. In MLD and ELD, there is no recovery of a packet loss because the recovery of the packet loss requires acknowledgement and re-transmission and causes scalable issue.

Multimedia application is generally not required re-transmission and particularly suitable for using the layered multicast. In the transfer of continuous data streams for multimedia application, delay constraints generally do not allow to give absolute guarantees on the integrity of the received data, and the recovery of lost packets is generally useless. The goal, for a continuous data stream, is to transfer data with the maximum achievable quality for the available bandwidth. This is simply achieved by using the layered data organization, where each subscription level corresponds to the same data transmitted with a different quality.

Moreover, when congestion occurs in MLD or ELD, the highest layer will be completely dropped. By doing this, we can protect the more important layer and drop out the less important layer so as to improve the overall multimedia quality.

Since receivers are expected to receive the same data for the multi-media transmission, it can save a lot of bandwidth by using IP multicast. With the layered multicast, each receiver can subscript with different level of layer which corresponds to the same data with a different quality based on the network conditions.

8. Conclusion and Future Work

We conclude this dissertation with a summary of our work. We then identify some research problems that can be addressed in the future work.

8.1 Conclusion

This dissertation presented a TCP friendly congestion control for layered multicast. We described the advantage of the layered multicast and mentioned that TCP fairness is one of the critical factors for successful deployment of layered multicast. The main objective is to establish a TCP friendly congestion control for layered multicast. In view of this, we initially investigated the TCP fairness and other aspects of the existing congestion controls:

- Priority dropping
- RLM
- RLC

And showed that none of them are TCP friendly. Then we incorporated the idea of priority dropping and AIMD property of TCP to establish the MLD and ELD congestion controls.

We examined both congestion controls through simulations. Our simulations revealed that both congestion controls did illustrate TCP fairness properties. Our results showed that both layered multicast and TCP could get the bandwidth allocation in a fair manner. Also we observed that the receivers with shorter RTT could get more bandwidth allocation than those receivers with longer RTT do. However, there is some tradeoff between TCP-fairness and intra-fairness.

Other than TCP fairness, we pointed out other important aspects of MLD and ELD, which are required further investigations. Finally, we discussed some of the implementation and deployment issues of ELD and MLD.

Basically, MLD and ELD are end to end congestion controls with router assistance for layered multicast. With the assistance from the routers, receivers can get more information to adopt the transmission rates to reflect the network condition. However, the tradeoff is that it complicates the implementation and deployment for layered multicast.

8.2 Future Work

In this section, we identify several new research problems as future work for this dissertation:

- TCP fairness under short-lived TCP connection: In our simulation, we use the long-live TCP flows (i.e. FTP traffic) to compare with the layered multicast. However, a reasonable portion of today's Internet traffic consists of short-live TCP connection (i.e Telnet, Wed). Therefore, it is also important to examine the TCP-fairness of the MLD and ELD in the presence of the realistic background traffic.

- Real world experiment: We should also consider to exam the MLD and ELD in a real network environment to validate the simulation results. The results will also help to identify some of the potential issue when we are going to deploy MLD and ELD in the Internet.

- Future simulation analysis for intra-protocol fairness and convergence time: We did some studies for these two properties for layered multicast. However, it is necessary to conduct an extensive simulation testing to validate our thinking. It is another key step to elaborate the MLD and ELD and make sure it can be deployed to Internet.

- Determine the RTT for MLD and ELD: In MLD and ELD, RTT is the important parameter to determine the join-timer. In this thesis, we indicate there are 4 different ways to estimate the RTT for a multicast group. Future analysis is necessary to find out the best way to get the RTT for MLD and ELD.

9. References

1. S. Deering, *Internet multicast routing: State of the art and open research issues*, Oct. 1993, Multimedia Integrated Conferencing for Europe (MICE) Seminar at the Swedish Institute of Computer Science, Stockholm.

2. S. Bajaj, L. Bresalu, and S. Shenker. *Uniform versus Priority Dropping for Layered Video*, ACM SIGCOMM'98 Conference; Applications, Technologies, Architectures and Protocols for Computer Communication; 1998 September 2-4; Vancouver, BC, Canada. pp. 131-143.

3. S. Floyd and K. Fall, *Router Mechanisms to Support End to End Congestion Control*, Network Research Group, Lawrence Berkeley National Laboratory, Berkeley CA, Feb 15, 1997, ftp://ftp.ee.lbl.gov/papers/collapse.ps.

4. S. McCanne, V. Jacobson, M. Vetterli, *Receiver-driver Layered Multicast*, SIG-COMM’96, August 1996, Standard, CA, pp. 1-14.

5. L. Vicisano, L. Rizzo, J. Crowcroft, *TCP-like congestion control for layered multicast data transfer*, Proceedings of IEEE INFOCOM98, San Francisco, CA, March, 1998.

6. J. Conlan, B. Whetten, *A Rate Based Congestion Control Scheme for Reliable Multicast*, Technical White Paper, Nov 24, 1998, GlobalCast Communication.

7. D. Rubenstein, J. Kurose, D. Towsley, *The impact of Multicast Layering on Network Fairness*, Department of Computer Science, University of Massachusetts at Amherst, February, 1999, UM-CS-1999-008.

8. T. Montgomery, *A loss tolerant rate controller for reliable multicast*. Technical Report NASA-IVV-97-011, West Virginia University, August 1997.

9. T. Sano, N. Yamanochi, T. Shiroshita, O. Takahashi, *Flow and congestion control for bulk reliable multicast protocols – toward coexistence with TCP*. RM meeting, Sep. 1997, http://info.isl.ntt.co.jp/chisho/rmtp/infocomm98.ps.gz.

10. V. Jacobson. *Congestion avoidance and control*. In Proc. SIGCOMM'88, page 314-329, 1988.