# Design and Performance Evaluation of Multicast Congestion Control for the Internet

By

**Somnuk Puangpronpitag**

**Submitted in accordance with the requirements for the degree of Doctor of Philosophy**


**THE UNIVERSITY OF LEEDS**
**SCHOOL OF COMPUTING**

November 2003

The candidate confirms that the work submitted is his own and that appropriate credit has been given where reference has been made to the work of others.

# Acknowledgements

I would like to acknowledge the following people, who have given me support and encouragement during this research:

Professor Roger Boyle (my supervisor) for his guidance, support and encouragement throughout the time of supervising this research. Many thanks are also for Dr. Karim Djemame (my advisor) and Dr. Mourad Kara (Energis) for sharing their knowledge and experience during this research.

Furthermore, I want to thank Mike Luby (Digital Fountain), Vivek Goyal (Digital Fountain), and Arnaud Legout (Castify Network) for their feedback and discussion about problems found during installation and validation of FLID-DL and PLM modules. I am also grateful to ns-2 developers for the solid simulation tools. In addition, I would like to thank ns-2 community for fruitful feedback on our work.

Many thanks go to John Kitching for his feedback on the early draft of this thesis.

I would also like to acknowledge my sponsors, Thai Royal Government and University of Mahasarakham, for granting my study leave and sponsoring my research. Special thanks go to Asst. Prof. Wirat Phongsiri for his support in cooperation with Mahasarakham during my academic leave.

My heartiest gratitude goes to my family members and friends (my father, my mother, my sisters, my brother, Raweeras Kongpitaksakul, and Priaopan Janesiriwanich), who have given me tremendous spiritual support. Without them, none of this would have been possible. Thank you.

# Abstract

Multi-rate Multicast Congestion Control (MR-MCC) is a promising opportunity to tackle the multicast congestion control problem in huge and heterogeneous networks like the global Internet. However, it is not easy to provide an MR-MCC design with responsiveness, efficiency of network utilisation, low packet loss, scalability and fairness (including inter-protocol fairness, intra-protocol fairness, intra-session fairness and TCP-friendliness) as well as feasible implementation.

This thesis is concerned with the design and performance evaluation of multi-rate multicast congestion control. We aim to address the problems faced by the previous proposals. In doing so, we have established a rigorous performance evaluation methodology via network simulation, and defined a set of key evaluation criteria to test MR-MCC protocols. Then, we have undertaken a performance evaluation of the previously proposed MR-MCC protocols (RLM, RLC, FLID-DL and PLM). Having learnt from our simulation analyses of previous proposals, we propose our innovative design of an experimental MR-MCC protocol, called Explicit Rate Adjustment (ERA). The design goals are scalability, responsiveness, fast convergence, fairness (including intra-session fairness, intra-protocol fairness, and inter-protocol fairness, in particular TCP friendliness), efficiency in network utilisation, and simplicity to implement. We have also implemented our experimental MR-MCC protocol in the ns-2 network simulation package. Through simulation, we demonstrate the performance evaluation of our MR-MCC extensively and demonstrate that it provides the desirable properties mentioned previously.

# Declarations

Some parts of the work presented in this thesis have been published in the following articles:

**S. Puangpronpitag, R.D. Boyle, and K. Djemame**, "Explicit Rate Adjustment: an Efficient Congestion Control Protocol for Layered Multicast", 11th IEEE International Conference on Networks, Sydney, Australia, September 2003.

**S. Puangpronpitag, R.D. Boyle, and S. Hassan**, "Explicit Rate Adjustment for Multi-rate Multicast Congestion Control using TCP throughput equation and Packet-pair Probe", 9th Asia Pacific Conference on Communications, Penang, Malaysia, September 2003.

**S. Puangpronpitag and R.D. Boyle,** "Performance Comparison of Explicit Rate Adjustment with other Multi-rate Multicast Congestion Control Protocols", 19th UK Performance Engineering Workshop, Warwick, UK, July 2003.

**S. Puangpronpitag, R.D. Boyle, and K. Djemame**, "Performance Evaluation of Layered Multicast Congestion Control Protocols: FLID-DL vs. PLM", International Symposium on Performance Evaluation of Computer and Telecommunication Systems, Montreal, Canada, July 2003. The extended version is available as Technical Report No. 2003-07, School of Computing, University of Leeds, UK.

**S. Puangpronpitag, M. Kara, and K. Djemame**, "ATM-GFR Buffer Management and Scheduling Issues: a Performance Study using TCP", 17th Annual UK Performance Engineering Workshop, Leeds, UK, July 2001.

**S. Puangpronpitag, M. Kara, and K. Djemame**, A Performance Evaluation of Buffer Management and Scheduling for ATM-GFR using TCP", 8th IFIP Workshop on Performance Modelling and Evaluation of ATM and IP Networks, Ilkley, UK, July 2000.

# Contents

# List of Tables

# List of Figures

# Abbreviations

| | |
|---|---|
| ACK | Acknowledgement |
| AES | Advance Encryption Standard |
| AIMD | Additive Increase Multiplicative Decrease |
| ALMA | Active Layered Multicast Adaptation |
| AN | Active Networks |
| ARQ | Automatic Repeat reQuest |
| ASM | Any-Source-Multicast |
| ATM | Asynchronous Transfer Mode |
| BE | Best Effort |
| BER | Bit Error Rate |
| CBR | Constant Bit Rate |
| CIFL | Coding Independent Fair Layered multicast |
| CSCW | Computer System Collaborative Working |
| DES | Data Encryption Standard |
| DF | Digital Fountain |
| DIDD | Doubling Increase Doubling Decrease |
| DL | Dynamic Layering |
| DoS | Denial-of-Service |
| DSL | Digital Line Service |
| DVMRP | Distance-Vector Multicast Routing Protocol |
| $E$ | Efficiency of Network Utilisation |
| ERA | Explicit Rate Adjustment |
| $F$ | TCP-Friendliness Ratio |
| FEC | Forward Error Correction |
| FDDI | Fibre Distributed Data Interface |
| FIFO | First In First Out |
| FLID | Fair Layer Increase Decrease |
| FLID-DL | Fair Layer Increase Decrease with Dynamic Layering |
| FPF | First Packet-pair Flag |
| FQ | Fair Queuing |

| | |
|---|---|
| FTP | File Transfer Protocol |
| GloMoSim | Global Mobile Simulator |
| *I* | Intra-protocol Fairness Index |
| ICMP | Internet Control Message Protocol |
| IETF | Internet Engineering Task Force |
| IGI | Initial Gap Increasing |
| IGMP | Internet Group Multicast Protocol |
| IP | Internet Protocol |
| IPSEC | Internet Protocol Security |
| IPv4 | Internet Protocol version 4 |
| IPv6 | Internet Protocol version 6 |
| IRTF | Internet Research Task Force |
| ISP | Internet Service Provider |
| LAN | Local Area Networks |
| LBNL | Lawrence Berkeley National Laboratory |
| LCC | Layered Congestion Control |
| LCT | Layered Coding Transport |
| LDA | Loss Delay Adaptation |
| LID | Layer Identifier |
| *LinRD* | Linearly Receiver Dependence |
| *LogRD* | Logarithmically Receiver Dependence |
| LT | Luby Transform |
| MCC | Multicast Congestion Control |
| MD5 | Message Digestive 5 |
| MLDA | Multicast enhanced Loss-Delay Adaptation |
| MR-MCC | Multi-rate Multicast Congestion Control |
| MSEC | Multicast Security |
| MTCP | Multicast TCP |
| NAK | Negative Acknowledgement |
| NDT | Network Diagnostic Tool |
| Ns | Network Simulator |
| OID | Object IDentifier |

| | |
|---|---|
| OSPF | Open Shortest Path Finding |
| OTcl | Object-oriented Tool Command Language |
| PDA | Personal Digital Assistant |
| PGMCC | Pragmatic Multicast Congestion Control |
| PIM-DM | Protocol Independent Multicast Dense Mode |
| PIM-SM | Protocol Independent Multicast Sparse Mode |
| PIM-SSM | Protocol Independent Multicast Source-Specific Mode |
| PLM | Packet-pair receiver-driven cumulative Layered Multicast |
| PLR | Packet Loss Ratio |
| PP | Packet-pair Probe |
| PSN | Packet Sequence Number |
| *Q* | Equality Index |
| QoS | Quality of Service |
| RAI | Rate Adaptation Interval |
| REAL | REalistic And Large |
| RED | Random Early Discard |
| REM | Random Early Marking |
| RFC | Request For Comment |
| *RI* | Receiver Independence |
| RIP | Routing Internet Protocol |
| RLC | Receiver Layered Congestion Control |
| RLM | Receiver Layered Multicast |
| RTT | Round Trip Time |
| RNG | Random Number Generator |
| SACK | Selective ACKnowledgement |
| SAM | Session Announcement Message |
| SCT | Sender Current Time |
| SP | Synchronisation Point |
| SR-MCC | Single-rate Multicast Congestion Control |
| SSM | Source-Specific Multicast |
| Tcl | Tool Command Language |
| TCP | Transmission Control Protocol |

| | |
|---|---|
| TFMCC | TCP Friendly Multicast Congestion Control |
| TG | Transmission Group |
| UCLA | University of California Los Angeles |
| VINT | Virtual InterNet Test-bed |
| WAN | Wide Area Networks |
| WEBRC | Wave and Equation Based Rate Control |
| WFQ | Weighted Fair Queuing |
| YATS | Yet Another Tiny Simulator |

# Chapter 1

# Introduction

## 1.1 Research Motivation

The rapid growth of the Internet has sparked the demand of several novel applications of group communication, such as Distributed Databases, Distributed Computing, Real Time Group Communication, Multiparty Videoconferencing, Media-on-demand Broadcast, and Bulk File Transfer Protocol (FTP) for software distribution. All these applications require an efficient distribution of data simultaneously to multiple receivers.

Multiple unicast transmission has been found inefficient to serve these applications since it can cause problems with network load. Therefore, an Internet multicast model (*IP Multicasting*) [47] has been introduced to deliver packets efficiently to groups of receivers. As shown in Figure 1-1, it allows the sender to send each packet just once on a network path. Then, the routers automatically forward the packet to each receiver that wants it. This multicast model helps minimise the number of copies of the packet that traverse the network.

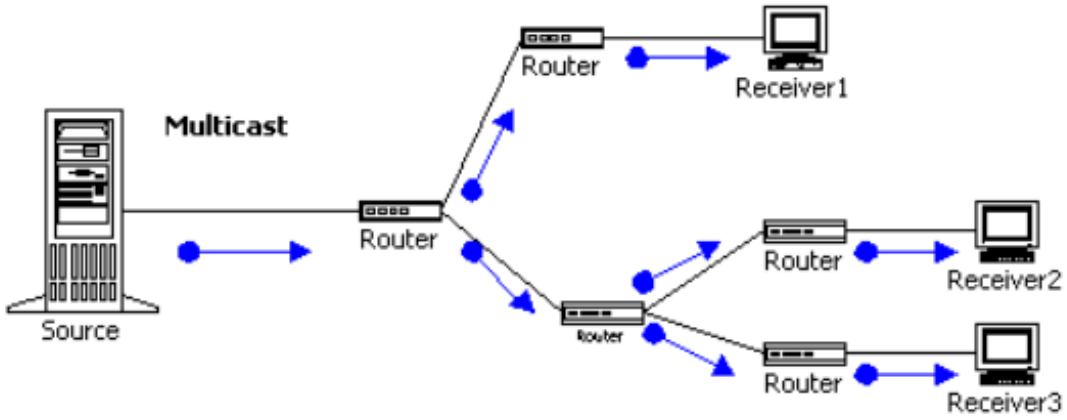



**Figure 1-1 Multicasting**

Although *IP Multicasting* has improved the efficiency of bandwidth usage in multiparty data delivery, it offers extremely limited support for congestion control. Without a congestion control mechanism, multicast applications in general can cause severe congestion-related damage to the Internet because a single multicast flow can be distributed throughout the Internet via a large global multicast tree [110]. Hence, a congestion control mechanism is one of the most important challenges in the widespread deployment of multicast.

Recently, several studies have tried to conquer this challenge. However, providing congestion control for multicast mode is far more difficult than providing it in unicast

mode. The design of multicast congestion control has to tackle several more problems, such as *Feedback Implosion*[1], rate adaptation for huge heterogeneous receivers, and *Round Trip Time (RTT)* estimation for multiple receivers.

So far, there have been two directions of multicast congestion control schemes proposed, namely *Single-rate Multicast Congestion Control (SR-MCC)* and *Multi-rate Multicast Congestion Control (MR-MCC)*. SR-MCC proposals (such as [50], [141], [147], [165]) have limitations in terms of scalability to a certain number of receivers only. So, it aims at networks other than the public Internet.

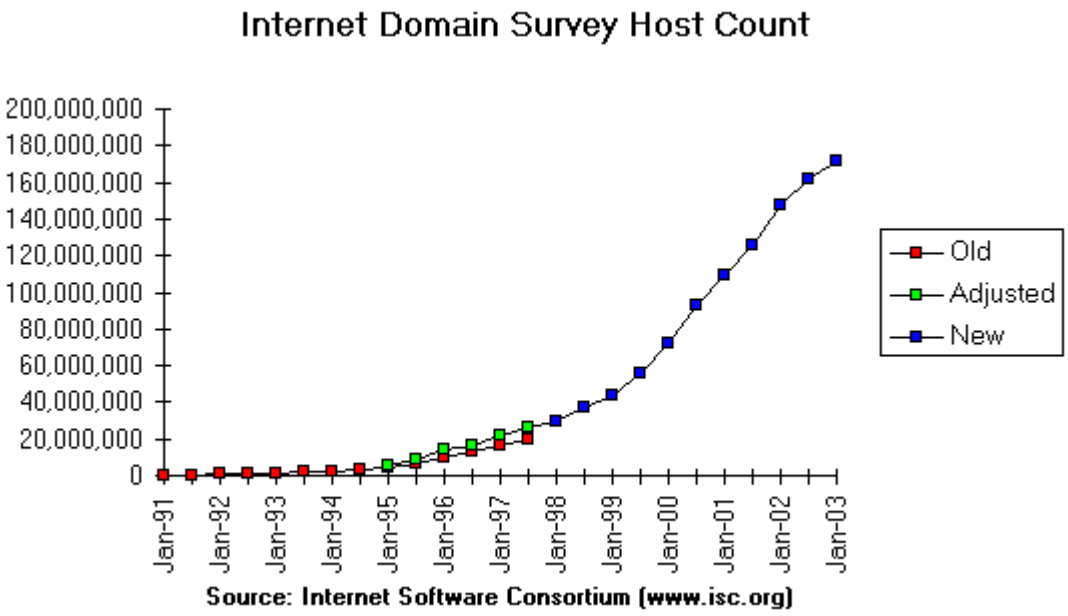



**Figure 1-2: Internet Growth *(Source: [2])***

The public Internet is a gigantic decentralised heterogeneous network interconnecting millions of all types of computing devices and networks throughout the world. The computing devices connected to the Internet can vary from desktop PCs, UNIX-based workstations, Personal Digital Assistants (PDA), TVs, mobile computers,

[1] The feedback implosion problem is explained further in Section 2.6.1.

automobiles, to even toasters [15], as well as other everyday domestic devices being connected. The networks connected to the Internet can range from dial-up modems, wireless, satellite, Digital Service Line (DSL) to high speed dedicated optical lines. In addition, as shown in Figure 1-2, the Internet keeps growing exponentially. In 1983, it comprised only 600 end hosts [2], [44]. It grew to 16-20 million end hosts in 1997 [2], [130], and 170-500 million end hosts as of January 2003 [2]. In order to provide multicast congestion control over the huge heterogeneous network environment, several MR-MCC proposals (such as [34], [95], [113], [159]) have been made.

Unlike classical congestion control schemes, MR-MCC introduces an innovative concept relying on two main techniques, namely *Layer Coding Transport (LCT)* [105] and *Receiver-driven Congestion Control*. So, it is also called *Receiver-driven Layered Multicast Congestion Control*. This new approach presents both opportunities and challenges. Several studies have recently been focused on the MR-MCC scheme, and various problems of MR-MCC have been revealed and solutions attempted. Nevertheless, there is still much room for research to be done before the dream of having multicast applications running over the Internet can become true. In particular, the question of how to provide MR-MCC with good responsiveness to network congestion, efficient bandwidth utilisation, high scalability, fairness towards the existing traffic, and implementation feasibility has not yet been answered.

## 1.2 Research Context

### 1.2.1 Scope of Research

Several studies on multicast communication are now in progress on different layers of the network protocol stack, such as network layer, transport layer and application layer. Some studies are interested in multicast error control. Some are interested in congestion control. This research will mainly focus on the transport layer only. Especially, the scope of the research covers only *Multicast Congestion Control (MCC)* problems, and in particular, this work is interested only in multicast congestion control for the Internet. Hence, we are only interested in MR-MCC protocols, not the SR-MCC ones.

### 1.2.2 Objectives

The work presented in this thesis focuses on optimisations of MR-MCC protocols. The goal is to yield greater performance in terms of responsiveness, efficiency of network utilisation, low packet loss, smoothness, scalability, feasibility, and TCP-friendliness (the fairness towards Transmission Control Protocol (TCP), which is a dominant traffic (representing more than 90% of traffic) on the Internet). In order to reach the goals stated, this work addresses several key areas as follows:

(1) A survey of existing studies of related research, in order to identify the problem areas

(2) A performance evaluation of existing MR-MCC proposals to explore their advantages and disadvantages in comparison, leading to a set of potential new solutions

(3) A design and simulation of an experimental MR-MCC protocol which deploys proven techniques to estimate available and TCP-friendly rate, together with our proposal of an innovative rate adaptation scheme and a new framework for the cooperation between the sender and receivers

(4) A performance comparison of the experimental MR-MCC protocol against existing MR-MCC proposals.

## 1.3 Key Contributions

This research presents a performance evaluation of previous MR-MCC proposals, and proposes a new design. By drawing upon the previous published work in this field, the initial contribution of this work is the establishment of a set of key performance evaluation criteria, simulation environment and performance metrics to evaluate MR-MCC protocols.

The further innovative contribution is a performance comparison through network simulation techniques of two recently proposed MR-MCC protocols (*Fair Layered Increase Decrease with Dynamic Layering (FLID-DL)* and *Packet-pair receiver-driven cumulative Layered Multicast (PLM)*). To the best of our knowledge, the performance comparison between these two protocols has never been done before.

Then, this study proposes a novel design of an experimental MR-MCC protocol that offers the following properties: scalability, responsiveness, fast convergence, fairness, efficiency in network utilisation, and feasibility to implement. Our design is based on an estimation of an explicit target rate using a proven technique for available bandwidth estimation of Keshav [86], and a TCP-friendly rate estimation technique of

Padhye [123]. Combining this estimation technique with the receiver-driven layered multicast approach of McCane [113] and our new framework for the cooperation between the sender and the receivers as well as our innovative rate adaptation scheme, we contribute an innovative experimental MR-MCC protocol, called *Explicit Rate Adjustment (ERA)*.

Finally, this study contributes a performance comparison between ERA (the new experimental MR-MCC protocol) with several previous proposals of MR-MCC protocols.

## 1.4 Thesis Outline

The remainder of this thesis is organised as follows:

**Chapter 2** discusses some introductory material and related background, including an overview of multicasting, the *Internet Multicast Model*, and multicast applications. The senses of fairness used in this thesis and the error control mechanism for multicast are also reviewed. Moreover, the network congestion problem is introduced. In particular, multicast congestion control is described in detail. Especially, MR-MCC proposals and problems are focused on.

**Chapter 3** presents the methodology used for this research. Alternatives to network performance evaluation techniques are presented. Particularly, this chapter justifies why network simulation techniques have been chosen as a key apparatus for this study. It also gives an overview of ns-2, a network simulator used for this study. Furthermore, the details of simulation construction, including defining simulation

objectives and performance metrics, specifying simulation model and environment, setting simulation parameters, processing the output data, and result analysis with confidence intervals, are included in this chapter.

**Chapter 4** presents a performance study of existing MR-MCC protocols. A strong review of existing MR-MCC proposals (such as *Receiver Layered Multicast (RLM)*, *Receiver Layered Congestion control (RLC)*, FLID-DL and PLM) has been given. Then, we establish a set of strong evaluation criteria and performance metrics to evaluate MR-MCC protocols. In particular, we choose to compare experimentally two most recent MR-MCC proposals – FLID-DL and PLM.

**Chapter 5** proposes a new design of a MR-MCC protocol. We name our experimental protocol "*Explicit Rate Adjustment (ERA)"*. The design goals are scalability, responsiveness, fast convergence, fairness (including intra-session fairness, intra-protocol fairness, and inter-protocol fairness, in particular TCP friendliness), efficiency in network utilisation, and simplicity to implement. In addition, an overview of security issues of our design is discussed.

**Chapter 6** presents a performance study of our new design and compares it with other existing MR-MCC protocols. This performance study is done by using the evaluation criteria and performance metrics defined in Chapter 4. It also checks whether the goals stated in Chapter 5 have been reached.

**Chapter 7** summarises this thesis and its contribution. Some suggestions for future work are also outlined.

# Chapter 2

# Background and Related Work

## 2.1 Introduction

This chapter provides the background of the related issues to be covered in this thesis. Following this section, Section 2.2 gives an overview of *Multicasting*, why it is important, *Internet Multicast Model*, and multicast applications. Network congestion problem is introduced in Section 2.3. In Section 2.4, we discuss multicast congestion control. The senses of fairness used in this thesis are described in Section 2.5. In section 2.6, we explain error control techniques for multicast protocols. Finally, in Section 2.7, the summary of this chapter is given.

## 2.2 Multicasting

### 2.2.1 Modes of Communication

We can broadly classify modes of communication into three, namely unicast, broadcast and multicast.

**Unicast**: In unicast mode, data packets are sent from one sender to one specific receiver (*one-to-one*). In this case, routers use a unicast routing protocol (such as Routing Internet Protocol (RIP) version 2 [108], Open Shortest Path Finding (OSPF) version 2 [116]) to establish the path between the sender and the receiver. Implementing unicast is simple. Yet, it is not scalable when the number of receivers increases. For sending to a huge group of receivers, multiple unicast could cause a problem with network load because it requires extra bandwidth for the same information even on shared links.

**Broadcast:** In broadcast mode, data packets are sent to every end hosts on the sub-network (*one-to-all*). The broadcast data packets are processed by every end host on the network, although some hosts may not be interested in the data packets. This could cause an unnecessary load on those end hosts and security problems of the data. According to [14], broadcast is difficult to route, especially over *Wide Area Networks (WAN)*. Generally, the network broadcasts only within the *Local Area Networks (LAN)* boundary to prevent *broadcast storms*. It is very difficult to broadcast across the Internet.

**Multicast:** In multicast mode, data packets are sent to a specific group of end hosts (*one-to-n*, when *n* is varied from zero to all). Multicast depends on the network to

forward the data packets to a specific group of end hosts that want them. In contrast with unicast, it helps reduce network traffic and the amount of processing that the sender has to do. Unlike broadcast, the multicast mode is not bounded only within LAN boundary. Yet, it can be used throughout the entire Internet [14].

To summarise, Table 2-1 shows the comparison of three communication modes.

| **Criteria** | **Unicast** | **Broadcast** | **Multicast** |
|---|---|---|---|
| Mode | *one-to-one* | *one-to-all* | *one-to-n* |
| Boundary | LAN, WAN | LAN | LAN, WAN |
| Scalability to receivers' size | No | Yes | Yes |

**Table 2-1: Three modes of communication**

### 2.2.2 Unicast vs. Multicast

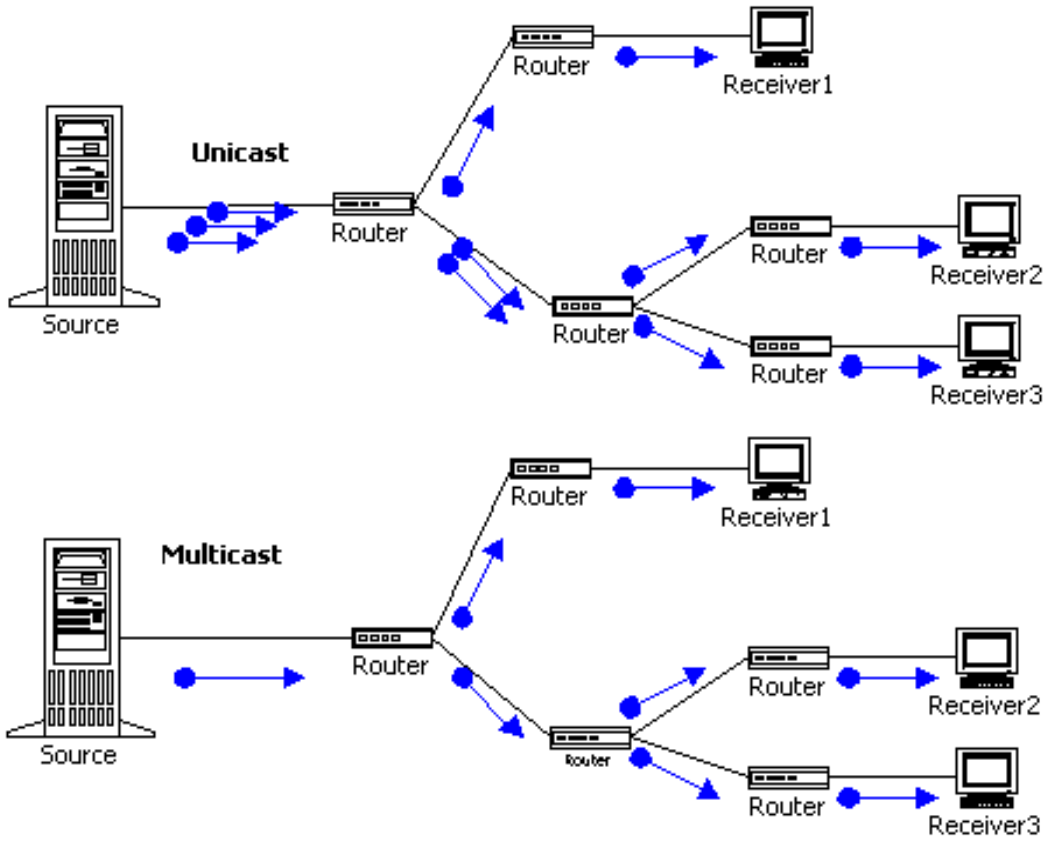



**Figure 2-1: Unicast vs. Multicast**

The comparison of transmission between unicast and multicast can be illustrated as shown in Figure 2-1. From the figure, unicast sends multiple copies of data packets (one copy for one receiver), while multicast sends only a copy of data packets to multiple receivers. Each data packet is sent just once on a network path; then the routers automatically forward it to each receiver.

As mentioned in Section 2.2.1, multicast is an efficient way to transmit data from a single sender to multiple receivers. Efficiency in multicast comes from: (1) the smaller number of transmissions that the sender has to process, and (2) the smaller number of data packets generated within the network.

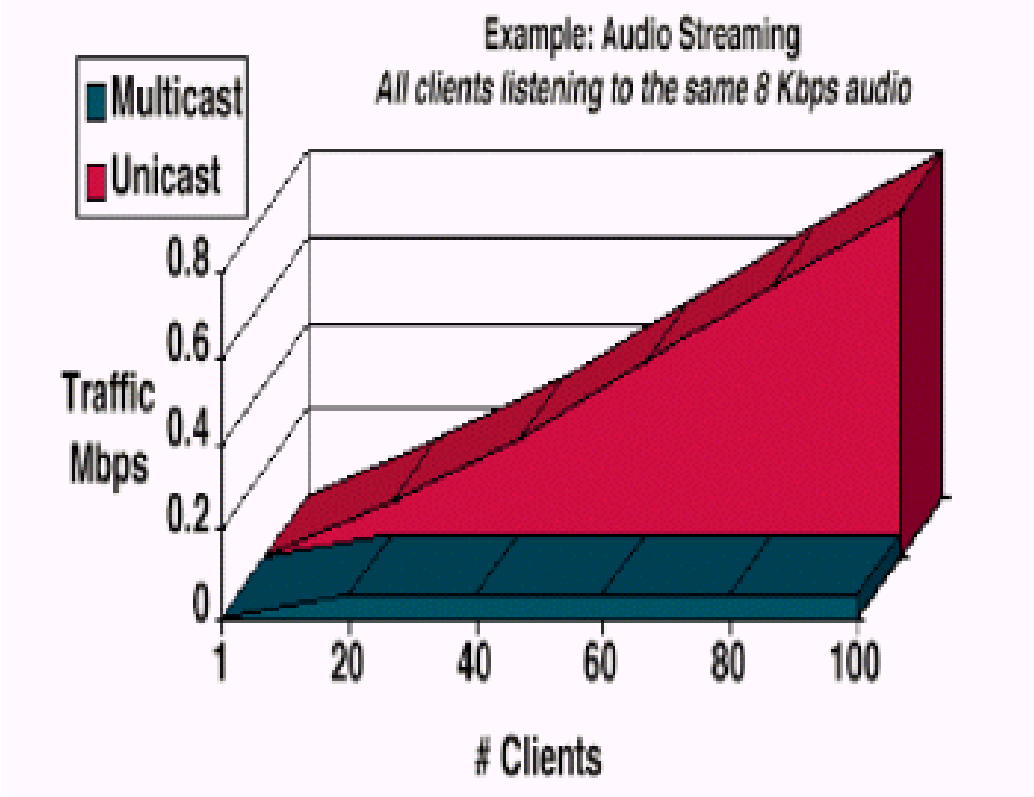



**Figure 2-2: Multicast traffic compared to unicast traffic** (*Source: [14]*)

To illustrate a comparison of network load between multicast and multiple unicast, we show an example by using the results (depicted in Figure 2-2) of an experiment by CISCO [14]. From the experiment, streams of an audio file at 8 Kbps were transmitted from one source to a number of clients (varying from 1 to 100) by multicast and multiple unicast consecutively. This example showed that multicast used much less bandwidth than multiple unicast when the number of receivers increased.

### 2.2.3 Internet Multicast Model

Deering [47] has introduced a model to support multiparty communication, called the *Internet Multicast Model* (also known as *IP Multicasting*) since 1989. It is a transmission of Internet Protocol (IP) datagrams to *an end-host group*, a set of zero or more hosts. This set of end hosts is identified by a single IP destination address, called

the *Multicast Address*. IP Multicasting is the same *Best Effort (BE)* delivery as the regular unicast IP, and also inherits all the robustness of the IP model.

The multicast addresses for *Internet Protocol version 4 (IPv4)* [134] are *IP address class D*, (i.e., those with "1110" as their high-order four bits). Those for *Internet Protocol version 6 (IPv6)* [48] use addresses with "1111 1111" as their high order eight bits. Figure 2-3 and Figure 2-4 show the multicast address for *IPv4* and *IPv6,* respectively.


**Figure 2-3: Format of multicast address for IPv4**

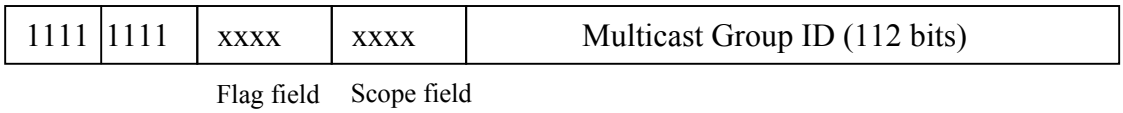



**Figure 2-4: Format of multicast address for IPv6**

For IPv4, IP multicasting uses the *Internet Group Management Protocol (IGMP)* ([35], [47], [58]) to handle group management dynamically. For IPv6, IGMP functions are incorporated into *Internet Control Message Protocol (ICMP).* The model is a very lightweight setup: senders just send, receivers announce interest, and the network delivers data to interested receivers [77].

Inter-network forwarding of IP multicast datagram is handled by *Multicast Routers.* The design of multicast routing algorithms for the multicast routers is one of the open research issues. There are a few basic algorithms for multicast routing identified

today, such as Distance-Vector Multicast Routing Protocol (DVMRP) [161], Protocol Independent Multicast Dense Mode (PIM-DM) [16], Protocol Independent Multicast Sparse Mode (PIM-SM) [150], and Protocol Independent Multicast Source Specific Mode (PIM-SSM) [27].

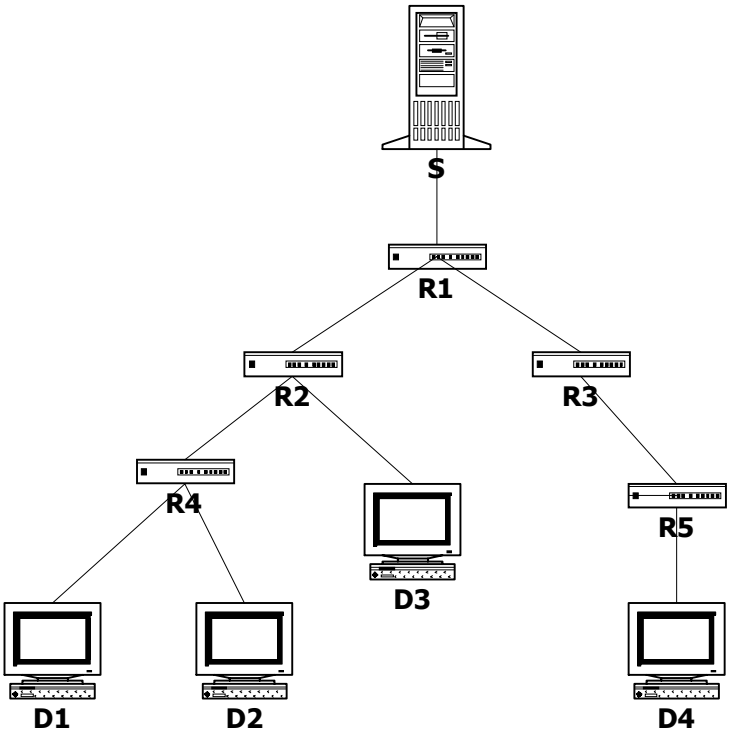



**Figure 2-5: Sample of the IP Multicasting**

Figure 2-5 illustrates an example of how the IP Multicasting model works. From the figure, the sender (*S*) multicasts data to the multicast group (*M*), denoted as the multicast session *(S, M)*. *S* simply uses the multicast address of *M* as its destination address. Without any receivers joining the multicast group *M*, the data from *(S, M)* are sent to the first multicast router (*R1*) only. There will be no further forwarding.

If a receiver, for example *D1*, wants to receive the data from this multicast session, it will join the multicast group (*M*) by simply sending an *IGMP JOIN message* for the multicast session *(S, M)* to its closest multicast router. Then, the multicast router *R4,*

which is the closest multicast router for this case, sends an IGMP JOIN message to the next level multicast router (*R2*) asking to join *M,* and *grafts* link *(R4, R2).* After that, *R2* sends an IGMP JOIN message to *R1,* and grafts link *(R2, R1)*. Hence, the multicast tree linking between *S* and *D1* is completely constructed. *D1* therefore can now receive the data from *S* for the multicast session *(S, M).*

Afterwards, if *D2* wants to join the session *(S, M)* as well, it sends an IGMP JOIN message to its closest multicast router as normal (which is also *R4*). In this case, *R4* just forwards the data on the link *(R4, D2)*.

Later on, if *D1* does not want to receive data from (*S, M)* anymore, it simply leaves *M* by sending an *IGMP LEAVE message* to its closest multicast router (*R4*). Since *D2* still joins *M, R4* only stops forwarding data to link *(R4, D1)*, but still keeps forwarding data to link *(R4, D2)*.

After that, if *D2* also leaves *(S, M), R4* must ensure that there are no other receivers active before stopping the flow of data packets by sending an *IGMP QUERY message* to poll the receivers in its subnet. If there is no response to stay receiving data from the multicast session *(S, M)*, *R4* notifies *R2* to *prune* multicast tree. *R2* then stops forwarding data to *R4.*

*R2* remains joining *(S, M)*. It would notify *R1* to prune multicast tree only when all receivers that it forwards data to have left the session. In this case, *R2* would not allow *R1* to prune multicast tree if *D3* still joins the session *(S, M)*.

### 2.2.4 Multicast Applications

**Variety of Application Requirements**

During the last decade, a lot of multicast applications have been emerging, such as distributed databases, distributed computing, real-time applications, multimedia applications, bulk-data transfer, video conference, and so on. The requirements of these applications are various in many aspects as follows:

- **Reliability:** They may require no reliability, full reliability (loss-sensitive), or partial reliability.
- **Packet ordering:** They may require packets to be sent *in-sequence* (*ordered*), or *out-of-sequence* (*non-ordered*).
- **Real-time:** Some applications are *real-time* (delay sensitive) while some are *time-shifted* (delay tolerant).
- **Jitter:** Some applications are jitter-sensitive but some are jitter-insensitive.
- **Distribution of communication:** Distribution of multicast applications can be one-to-many or many-to-many.

Table 2-2 shows some examples of multicast applications with their broad-range of requirements. The current transport protocols (such as TCP and User Datagram Protocol (UDP) [133]) are not enough to support these various requirements. This is because they provide only a simple interface to all applications. They offer only *"All or Nothing"*, i.e., TCP supports *ordered* and *reliable* applications while UDP supports *non-ordered* and *non-reliable* applications. This problem is known, as *"One size does not fit all"*. So, the multicast transport protocols need to be specified to support these various requirements of applications.

| Requirement | Application Examples |
|---|---|
| In-sequence + Reliable + Non real-time | FTP and other TCP-style applications |
| In-sequence + Reliable + Delay-sensitive | Financial and stock exchange applications |
| Out-of-sequence + Partially Reliable + Delay-sensitive | Video player application |
| Out-of-sequence + Reliable + Non real-time | Image server |
| Out-of-sequence + Non-reliable + Non real-time | UDP-style Applications |
| In-sequence + Partially reliable + Delay-sensitive + Jitter-sensitive | Audio and video conference |

**Table 2-2: Various requirements of multicast applications**

**Two Categories of Multicast Applications**

We can classify multicast applications into two groups, namely reliable multicast applications and multimedia multicast applications. Some examples of reliable multicast applications are Distributed Databases, Financial and Stock Exchange Information Distribution and other *Content Delivery Applications*. These applications need data reliability. They are *loss-sensitive*, and need error control mechanisms to ensure a remedy of packet loss.

**Multimedia Applications**

According to Kurose [89], multimedia applications can be categorised into three classes, namely *Streaming Stored Multimedia*, *Streaming Live Multimedia*, and *Real-time Interactive Multimedia*.

For streaming stored multimedia applications, their contents (such as audio or video files) are pre-recorded and stored at a server. A user can request these contents on-demand. The user may pause, rewind, and fast-forward through the multimedia contents. To play the contents, the user has to spend a few seconds downloading part of contents (called *Stream*). The contents then are downloaded and played streams after streams. This streaming technique is to avoid a long delay of downloading the entire file.

For streaming live multimedia applications, they allow a user to receive a *live* audio and video broadcast. The streaming live audio/video is not stored on any server. Hence, the user cannot fast-forward through the contents. Yet, by locally storing the *content streams,* the user can still pause and rewind the contents. The first audio multicast started officially on March 1992 [38], while the first distribution of unicast audio was in the 1970s [163].

Real Player™ of Real Networks [8], Quick Time™ [9] of Apple and Windows Media Player™ [10] of Microsoft are examples of the (live and stored) streaming multimedia tools. Distribution of live multimedia contents can be efficiently accomplished by multicasting. However, until the time of writing this thesis, live multimedia contents are still mainly distributed via multiple unicast [89]. The reasons behind the multicasting un-deployment are discussed in the next Section.

Real-time interactive multimedia applications includes *Internet Phone* [4], *Videoconferencing,* and *Computer System Collaborative Working (CSCW)*, also known as *GroupWare*. For *Internet Phone* applications, they may be PC-to-PC

Internet phone, PC-to-phone, or phone-to-Internet-to-phone. *VocalTec Internet Phone*™ [11] and *Microsoft Net Meeting*™ are examples of the real-time interactive multimedia tools

### 2.2.5 Obstacles to Widespread Adoption of Multicast

Multicast is a better transmission mode in sending to multiple receivers. It is also now supported by major router manufacturers. The Internet therefore becomes increasingly multicast capable [96]. However, the IP multicast model (proposed since 1988) has not been widely deployed on the Internet until now. Most Internet Service Providers (ISP) do not turn the multicast option on [89], [96]. According to [52] and [96], this may result from: (1) lack of suitable multicast congestion control algorithms, (2) no incentive for any flows to use multicast instead of multiple unicast, (3) overhead of multicast (such as IGMP), (4) security concern, (5) difficulties of network management, and (6) lack of valid charging model. In this research, we try to tackle the first problem.

## 2.3 Network Congestion

### 2.3.1 Definition of Network Congestion

Legout and Biersack [93] give the definition of congestion as follows:

"*A network is said to be congested from the perspective of user i if the satisfaction of i decreases due to a modification of the characteristics of his/her network connection.*"

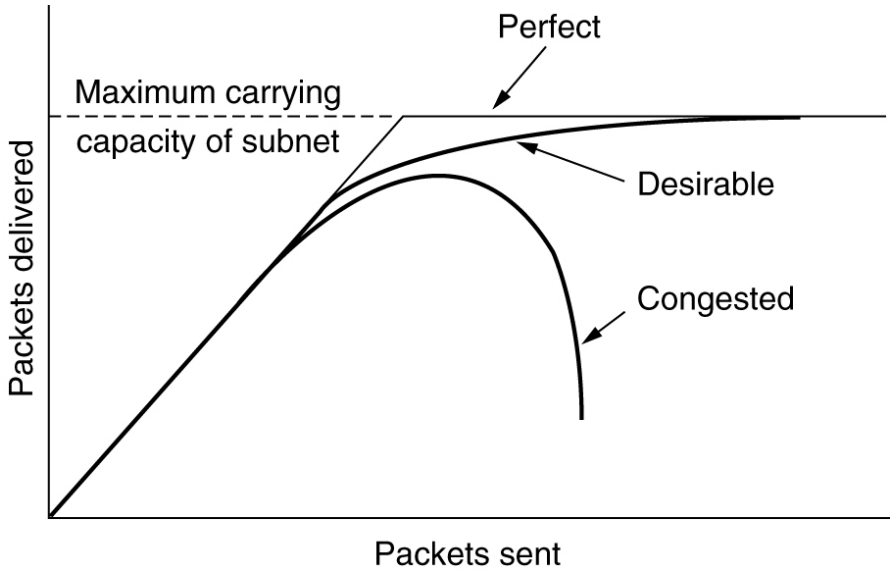



**Figure 2-6: Network Congestion** *(Source: [155])*

According to [155], network congestion happens when too many packets are sent to network (beyond the capacity of the network) and the performance (measuring from the number of packets delivered) drops. From Figure 2-6, when the network is not yet congested, the number of packets sent and received is the same except for a little packet loss due to transmission error. Later, when the load (the number of packets sent) is beyond the network capacity, the routers are no longer able to cope, and begin dropping packets. In this case, network congestion has occurred. The number of packets received can keep decreasing if there is no congestion control mechanism to recover the situation. In the worst case, *Congestion Collapse* can happen.

### 2.3.2 Congestion Collapse

Congestion collapse (also known as *the Internet Melt Down*) was first identified by Nagle in 1984 [117], which he defined as "*a drastic drop in the available bandwidth because of an unsociable transport protocol sending more than which a network can handle, thus rendering the network unusable*".

The Internet's congestion collapse was first observed on October 1986, where data throughput on a link in UC Berkeley dropped from 32 Kbps to 40 bps. This resulted from congestion control algorithms that were not adequate in the original version of TCP [75]. From then, the congestion control algorithms (*Slow-start* and *Congestion Avoidance* [75] were introduced into the TCP protocol to prevent this kind of disaster from reoccurring.

### 2.3.3 Congestion Control

ISP 's network links have limited capacity, while there is users' insatiable demand for bandwidth. Moreover, an expeditious emergence of multimedia and multicast applications (such as distributed computing, audio, video, and interactive online games), driven by broadband access technologies, has intensified the network congestion problem. According to Jain [79], network congestion problems cannot be solved by increasing additional bandwidth. No matter how much bandwidth is added, networks can still face a congestion problem. This is because the network congestion problem is not a *static resource shortage problem*, but a *dynamic resource allocation problem*. Increasing additional bandwidth just circumvents the problem without providing a real treatment. Furthermore, it is cost-prohibitive (due to requiring upgrades to network equipment) to make networks handle peak loads that only occur

for a few hours a day. So, congestion control mechanisms are required to tackle congestion problems.

TCP is a dominant congestion control mechanism for the unicast. Yet, for multicast, the congestion control research is still at an infant stage. In this research, we focus only on multicast congestion control.

## 2.4 Multicast Congestion Control

The effect of congestion problems in multicast would be more dangerous to the Internet. This is because a single multicast flow may be distributed along a large multicast tree reaching throughout an enormous part of the Internet. Compared with unicast, multicast congestion control is also more difficult to provide. There are a lot of problems, such as:

- There are potentially different bottleneck bandwidths from multiple receivers to the source. Then, it can be difficult to define a target data rate.
- Feedback on congestion is difficult. If the feedback is too sluggish, the congestion can persist. If the feedback is too aggressive, *Feedback Implosion*[2] [22] can happen.

In recent years, several multicast congestion control protocols have been proposed. They can be classified into single-rate and multi-rate multicast congestion control schemes.

[2] The feedback implosion problem is explained further in Section 2.6.1.

### 2.4.1 Single-Rate Multicast Congestion Control (SR-MCC)

With a single-rate multicast congestion control scheme, all receivers in a multicast session receive the data at the same reception rate. The scheme picks one or more of the slowest receiver(s) as *Representative(s)* (also called *Acker(s)* or *Sender Agent(s)*). Then, the sender adapts the transmission rate to the *Representative(s)*. Pseudofed [50], Multicast TCP (MTCP) [141], Pragmatic Multicast Congestion Control (PGMCC) [147] and TCP-friendly Multicast Congestion Control (TFMCC) [166] are examples of single-rate multicast congestion control protocols.

SR-MCC's weak point is that all receivers receive packets at the same rate, which is the worst-case rate. A single sluggish receiver can retard the reception rate of all other receivers that may be in better network conditions. When there are more receivers joining the session, the reception rate of each individual receiver has a tendency to degrade. This can lead to the *Drop-to-Zero Problem* [162], where the transmission could stall after a certain period of time due to the over-pessimistic wrong estimation of network congestion at the sender. In order to prevent all receivers suffering from having a very slow receiver, rules would be set to detect and force such a receiver to unsubscribe. Hence, fundamentally, the SR-MCC scheme has significant limitations in handling a very large heterogeneous group of receivers. For example, Seada and Helmy have shown in [149] that PGMCC exhibits poor performance and unfairness when having very heterogeneous receivers. In summary, SR-MCC is intended only for *Intranet*, *Extranet*, or *Managed Networks* other than the public Internet.

### 2.4.2 Multi-Rate Multicast Congestion Control (MR-MCC)

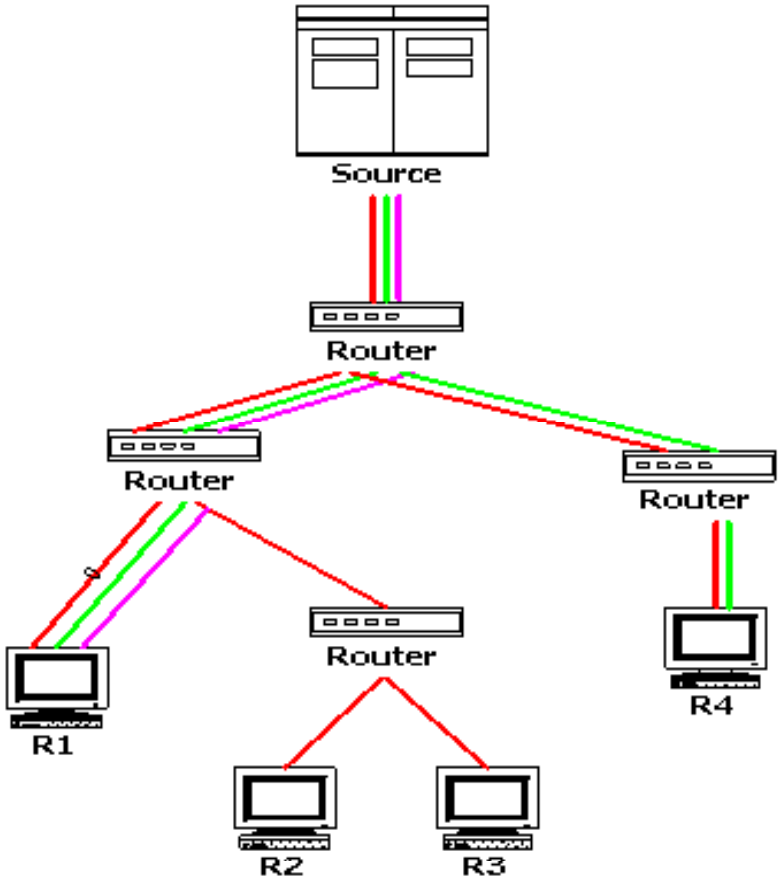



**Figure 2-7: Multi-rate multicast congestion control**

The multi-rate multicast congestion control scheme is more flexible in bandwidth allocation along different network paths. It allows the various receivers, in a single session, to receive data at different reception rates. As shown in Figure 2-7, this scheme is based on the ability of a source to generate the same data at different rates over multiple multicast streams (using *Layer Coding Transport (LCT)* [105]). The multiple multicast groups are organised into logical layers. Each layer is a multicast group.

For multimedia applications (such as video broadcasting), encoding is done in such a way that each layer has the same content, but with an increasing quality. The base

layer contains the obligatory data for decoding. The extended layers provide more information. This information when combined with the base layer's information results in improved quality. The more layers subscribed, the more bandwidth is consumed, and the better quality received.

For reliable content delivery applications (such as bulk data transfer), each layer is encoded by using Forward Error Correction (FEC) ([104], [106]). This is to provide reliability for data transport (see also Section 2.6.2). In this case, the base layer provides the crucial data for decoding, while the extended layers provide the increased rates to reduce the overall transfer time. The more layers subscribed, the more bandwidth consumed, and the quicker the transfer rate gained.

The burden of congestion control is at the receiver's side. The source has only a passive role, so this approach is called *receiver-driven*. A receiver tunes its reception rate by subscribing to and unsubscribing from layers according to its network condition. Each receiver takes its own congestion control decision autonomously. So, different receivers may subscribe at different levels of their own subscription rates, especially when they are in different network conditions. Basically, the receiver starts from subscribing to the base layer, and then decides whether it can subscribe to the next layer. The decision is made by its network conditions, possibly upon several factors (such as packet loss, or multicast round trip time) up to its design. For the first proposal of MR-MCC (RLM), packet loss is used for the decision, while the protocol introduced in this thesis uses an explicit estimated target rate instead. Their details and comparison will be discussed later.

The advantages of this scheme are: (1) it is massively scalable to a high number of receivers because the sender's behaviour is independent of the number of receivers, and there is no feedback from receivers; (2) the reception rate is not bound to the slowest receiver like SR-MCC, thus it can provide higher throughput to each individual receiver; (3) there is no *feedback implosion* problem, since no feedback is needed from receivers.

**IGMP Leave Latency Problem**

Congestion control in MR-MCC is performed indirectly by the group management and multicast routing protocols at the network layer. It relies on the ability of multicast routers to quickly start/stop distribution of a multicast group on demand. Subscribing to a layer is joining the multicast group of that layer using IGMP, and grafting the multicast routing tree by using the IP multicast routing protocol. Unsubscribing from a layer is leaving the multicast group of that layer using IGMP, and pruning the multicast routing tree by using the IP multicast routing protocol.

When receivers try to tackle congestion by unsubscribing from a layer, it can take several seconds for this to take effect, due to the nature of IGMP. The IP Multicasting model uses IGMP between receivers and routers to maintain the membership information of a multicast group. This membership information is then used by the multicast routing protocol to prune and graft the multicast tree so that only the group members are reached by the multicast flow.

In leaving a multicast group, the receiver sends an *IGMP LEAVE* message to the router. Then, the router must ensure that there are no other receivers active before

stopping the flow of packets by sending an *IGMP QUERY* message to poll the receivers in its subnet. According to RFC-2236 [58], the router typically polls up to three times before terminating flow to the group. This is to ensure reliability; otherwise, the *QUERY* messages may be lost. Each polling attempt can take up to three seconds; thus IGMP leave delay can be up to nine seconds before successfully pruning the multicast tree [34]. During this latency, multicast traffic still flows through the router. So, the IGMP leave latency can slow down bandwidth recovery, and therefore causes congestion persistence. This problem is called the *IGMP Leave Latency Problem.*

There are two ways to tackle this problem. The first way is introducing *Dynamic Layering (DL)* to MR-MCC mechanisms. The other way is modifying IGMP protocol to be quicker in leaving multicast group. The detailed discussion of both solutions is provided in Chapters 4 and 5.

### 2.4.3 Some Established MR-MCC Protocols

**Receiver-driven Layered Multicast (RLM)**

RLM [113] is the first proposal of receiver-driven layered multicast congestion control by McCanne et al. It introduces the technique called *Join Experiment* to adjust receivers' reception rate to the network conditions. This involves the receiver increasing its reception rate by periodically subscribing to an additional layer, and decreasing its reception rate by unsubscribing from a layer when experiencing packet loss. Further details of RLM can be found in Chapter 4.

**Receiver-driven Layered Congestion Control (RLC)**

RLC [159] has been proposed to address some of the problems in RLM. It introduces *Forward Error Correction (FEC) encoding* for reliable multicast applications. This enables receiver-driven layered multicast to be applied not only to multimedia applications but also to reliable multicast applications like database replication or bulk data transfer. RLC introduces *Synchronised Join Experiments* to coordinate the downstream receivers. In addition, RLC also introduces the concept of *Burst Test* to avoid over-subscription. Further details of RLC are discussed in Chapter 4.

**Fair Layer Increase Decrease with Dynamic Layering (FLID-DL)**

FLID-DL has been proposed by Byers et al. [34] to address some deficiencies of RLC. FLID-DL still maintains the concept of *Join Experiment* and *Synchronisation Points (SP)* of RLC. It introduces two new techniques - *Dynamic Layering (DL)* and *Fair Layer Increase Decrease (FLID)*. FLID generalises the layering scheme and reduces the possibility of packet loss. DL mechanism helps mitigate the problem of

IGMP leave latency (See also Section 2.4.2). Further details of FLID-DL are given in Chapter 4.

**Coding Independent Fair Layered multicast (CIFL)**

CIFL has also been developed from RLM and RLC by Khayat and Leduc [88]. It maintains the concepts of join experiment and SP. However, its variant version of join experiment is enhanced with a *failure-learning algorithm* to reduce the over-subscription. To provide TCP-friendliness, CIFL rate adaptation is designed by integrating the TCP throughput equation of Mahdavi [112].

**Packet-pair receiver-driven cumulative Layered Multicast (PLM)**

PLM [95] is designed using the *Packet-pair Probe (PP)* approach to infer the available bandwidth and avoid congestion instead of relying on a join experiment technique like RLM, RLC, CIFL and FLID-DL. PLM also requires *Fair Queuing (FQ)* [51] at every router to provide TCP-friendliness. Further details of PLM are given in Chapter 4.

**Multicast enhanced Loss-Delay Adaptation (MLDA)**

MLDA (proposed by Sisalem and Wolisz [151]) uses the Loss Delay Adaptation (LDA) algorithms. To provide TCP-friendliness, MLDA adjusts the rate to the TCP throughput equation of Padhye [123]. MLDA also proposes to have feedback from receivers to the sender. The receivers report the rate to the sender, avoiding feedback implosion by using exponentially distributed timer suppression. Then, the sender continuously adjusts the bandwidth distribution of the layers to suit the reported rates.

**Active Layered Multicast Adaptation (ALMA)**

ALMA [169] has been proposed by Yamamoto and Leduc, using the *Active Networks (AN) Paradigm* [118] and a price function. The active capsules are used in ALMA to provide information to deal with congestion control.

**Wave and Equation Based Rate control (WEBRC)**

WEBRC (proposed by Luby et al. [102]) is designed using the wave-like scheme. Instead of having constant bit rate streams like other MR-MCC, WEBRC has periodic streams (called waves) with exponentially decreasing rate. This is to provide dynamic layering to tackle the IGMP leave latency problem.

## 2.5 Fairness

When several connections compete to share network bandwidth, the aspect of fair resource distribution needs to be considered. It is desirable for a network to offer its resources to the competing connections as fairly as possible. With a poor bandwidth distribution scheme, even for a high total network utilisation, some connections may enjoy a greater share of the resources at the expense of other connections.

### 2.5.1 Definition of Fairness

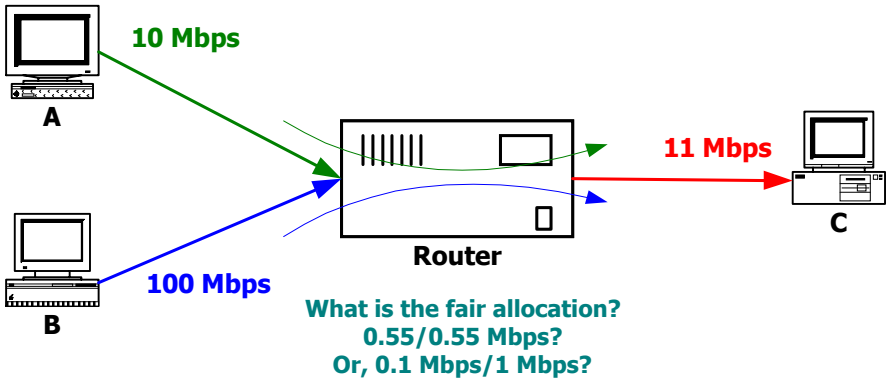



**Figure 2-8: Fairness Definition**

It is not easy to define fairness. As shown in Figure 2-8, fairness definition can be various. So far, a number of definitions have been proposed, such as *Equality Index* [81], *Max-min Fairness* [78], and *Proportional Fairness* [84]. However, up till now, there has been no consensus on the fairness definition. It is not a purpose of this thesis to give a fairness definition or compare the proposed ones. We only explain the details of equality index (which is used as our metric) in Chapter 4, and describe some senses of fairness (used in this work) in the next Section.

### 2.5.2 Senses of Fairness Considered in Our Experiments

As shown in Figure 2-9, we consider the following senses of fairness as our criteria to evaluate MR-MCC protocols (in Chapters 4 and 6):

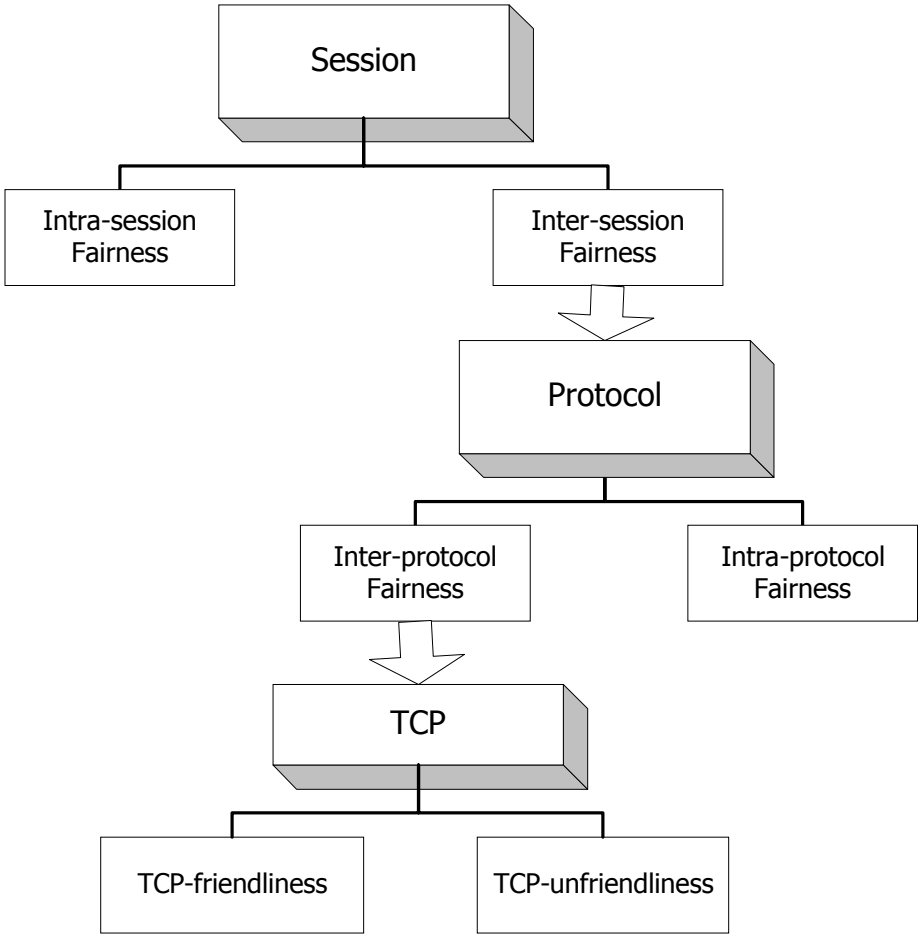



**Figure 2-9: Senses of fairness considered in this research**

**Intra-session fairness**

Intra-session fairness is fairness among receivers of the same multicast session. When they are under the same bottleneck link, they would be able to get the same utilisation.

**Intra-protocol fairness**

When several connections of the same congestion control protocol compete for bandwidth, they should be able to share it fairly.

**Inter-protocol fairness**

When several connections of different protocols compete for bandwidth, they should be able to share it fairly. A good congestion control protocol should be able to compete fairly with existing *Flow Adaptive Protocols*[3] (such as TCP). In particular, inter-protocol fairness towards TCP is very important, and is called *TCP-friendliness*.

### 2.5.3 TCP-friendliness

TCP traffic represents the majority of today's Internet traffic with more than 90% of the whole [42], [73], [156]. To protect the majority population from aggressive traffic, the *Internet Engineering Task Force (IETF)*[4] ([29], [59], [110]) suggests that new congestion control mechanisms for traffic likely to compete with *best-effort* TCP traffic should be TCP-friendly.

TCP-friendliness (also known as *TCP-compatibility*) is actually the inter-protocol fairness towards TCP. The TCP-friendliness definition [107] demands that a TCP flow and a non-TCP flow should receive similar steady-state bandwidth shares, if they have similar transmission behaviour (i.e., traverse the same path and thus face similar round trip delays and loss rates).

However, multimedia traffic and TCP are different in nature. Multimedia applications prefer congestion control mechanisms that respond more slowly but are less

[3] A flow adaptive protocol is a good behaviour protocol, which is responsive to congestion.

[4] The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet (http://www.ietf.org)

oscillatory than TCP. This is to produce a smoother bandwidth usage that brings smoother reception quality to users. So, it is expected that a TCP-friendly multimedia flow would acquire the same bandwidth share as a TCP connection, only averaged over time intervals of several seconds or even only over the entire lifetime of the flow [151]. We also note that some multimedia applications may need to have *Quality of Service (QoS)* guarantee instead of being left in the best-effort network environment. So, multimedia traffic flows in this context would have minimum bandwidth requirement, and may not be TCP-friendly anymore.

In addition, several studies (such as [45] and [93]) argue that TCP-friendliness paradigm does not extend to the new applications. According to the *TCP throughput equation* (see also Chapter 5, Section 5.5.8), TCP throughput heavily decreases in condition to packet loss rate. This behaviour does not fit all applications' requirement. Unlike TCP (which is loss-sensitive), some applications (such as audio, video, and other multimedia applications) are loss-tolerant but delay-sensitive. Their throughput would not be sensitive to packet loss rate as much as TCP would. Since some trivial loss can be tolerated by the applications, they would not decrease the throughput due to that loss. By using TCP-friendly congestion control scheme, they would therefore gain poor performance. Furthermore, the future Internet may not be dominated by TCP traffic anymore. Hence, it is doubtful whether the TCP-friendliness paradigm is a good fairness paradigm in the long term.

Although TCP-friendliness is a paradigm forced by IETF for any protocols to be deployed over the Internet and competing with TCP, the multicast vs. unicast bandwidth allocation is still an arguable issue. TCP is only a unicast protocol. Should

a multicast session be treated as a single session deserving no more bandwidth than a single TCP (unicast) session? A session of MR-MCC may serve several receivers, while a TCP session serves only one. In a common sense, the multicast session should therefore be given more bandwidth than TCP (unicast) connection. Especially, this is to give incentives in deploying multicast instead of *multiple unicast*, thus saving network load. Without gaining any more bandwidth share, the overhead of multicast (such as IGMP, complex security issues and difficulties in congestion control) would not be attractive. Then, most receivers would prefer to use multiple unicast, and not move on to multicast.

There have been a few initial studies focusing on defining a better fairness paradigm between multicast and unicast. For example, Wang et al. [162] have defined *Bounded Fairness* or *Essential Fairness* (against the current paradigm, *Absolute Fairness),* as the situation in which bandwidth allocated for multicast may not be the same as bandwidth allocated for unicast. The multicast flow in this sense of fairness may be allocated more bandwidth than TCP at a certain boundary. Furthermore, Legout et al. [96] proposed that the bandwidth allocation for multicast should consider the number of downstream receivers. They demonstrated three possible schemes of bandwidth allocation, namely *Receiver Independence (RI), Linearly Receiver Dependence (LinRD)* and *Logarithmically Receiver Dependence (LogRD)*. They have also shown that LogRD (which weights the number of downstream receivers by a logarithmic function) is the best among three. It can give more bandwidth allocated to multicast without starving unicast.

However, it is beyond the scope of this research to define a new paradigm of fairness between multicast and unicast. In our design and evaluation of MR-MCC, we decide to follow the IETF-enforced TCP-friendliness paradigm.

## 2.6 Error Control for Multicast Protocols

To support the multicast reliable applications, mechanisms to provide reliability of data transmission is needed. This is called *Error Control*. The function of error control is providing detection and remedy of lost packets. There are two directions of error control using for reliable multicast - *Automatic Repeat reQuest (ARQ)* and *Forward Error Correction (FEC)*.

### 2.6.1 Automatic Repeat reQuest (ARQ)

ARQ is a classical error detection and retransmission mechanism. This mechanism is very well known and works well for unicast reliable protocols, as evidenced by the success of TCP.

ARQ relies on the feedback (positive or negative acknowledgement) from the receivers to detect the transmission error. If a transmission error is detected, it automatically requests retransmission.

There are two main directions of the error detection for ARQ, namely sender-initiated reliable and receiver-initiated reliable. Sender-initiated reliable multicast protocols are based on the use of positive *Acknowledgements (ACK)* while receiver-initiated reliable multicast protocols are based on the use of *Negative Acknowledgements (NAK)*. Table 2-3 shows a comparison of these two approaches.

| Issues to compare | Sender-initiated reliable | Receiver-initiated reliable |
|---|---|---|
| Who is responsible for loss detection and recovery? | Sender | Receiver |
| Way of ensuring reliability | Receivers ACK correctly received packets | Receivers NAK lost packets. |
| Advantage | Easy resource management | Better scalability due to no state of receivers needed to be maintained at sender |
| Disadvantage and Problem | ACK Implosion | NAK Implosion. |

**Table 2-3: Sender-initiated reliable vs. Receiver-initiated reliable**

The steps of the sender-initiated reliable approach can be described as follows:

1. The sender transmits the packet and starts a timer at the time of a packet transmission.
2. The sender waits for an ACK from the receiver, providing that each time a receiver correctly receives a packet, it responds by sending an ACK.
3. The sender maintains the ACK list for each packet from the receivers. On the receipt of the ACK, the sender updates the ACK list for the corresponding packet.
4. If the sender receives ACKs for the data packet from all receivers before timeout, it reinitiates the ACK list. Then, go to step (1) to transmit the next packet.
5. If the time is out, a lost packet is assumed. Then, the sender retransmits the packet, and restarts the timer, as well as reinitiates the ACK-list.

The general steps of the receiver-initiated reliable approach can be illustrated as follows:

1. The sender continuously transmits packets.

2. When a receiver detects lost packets by noticing gaps in the sequence number of the packet stream, it sends a NAK to inform the sender of the non-receipt of the packet.
3. To protect either the loss of the NAK or the subsequent packet retransmission, the receiver starts a NAK timer in the same way that the sender does for ACK in a sender-initiated reliable approach.
4. On receiving a NAK, the sender retransmits the lost packet.

Several studies (such as [26], [39] and [125]) have focused on the sender-initiated reliable approach. [125] and [158] have tried to optimise this approach by grouping ACKs for different packets into a single control packet. However, [57] and [132] have later suggested that the receiver-initiated reliable approach using NAKs can achieve higher throughput performance than the server-based initiated reliable approach using ACKs for multicast transmission. In addition, the receiver-based initiated reliable approach would also scale better, since the sender does not need to maintain the states of receivers. Hence, the receiver-initiated reliable approach would be the preferable error detection scheme to implement reliable multicast protocols.

However, ARQ (even by receiver-initiated reliable approach) does not work well for one-to-very-many reliable protocols. This is mainly because of the *Feedback and Recovery implosion*.

The *feedback and recovery implosion* is a well-known problem of multicasting revealed in [22] and [57]. When the multicast session scales to a tremendous number of receivers, there could be a tremendous amount of feedback returning from the huge

number of receivers to the sender. With the huge number of receivers, the probability of packet losses in the receivers at any moment can also become very high. Consequently, the sender needs to perform retransmission persistently, and finally this results in a sharp increase of bandwidth consumption and more severe congestion and losses. Holbrook et al. [71] have discussed this problem as *Crying Baby*, in which a lossy receiver can provoke repetitive retransmissions and slacken the multicast session even for the less lossy receivers. Some of the proposed solutions to this problem are as follows:

(1) Feedback suppression by random timer and back-off techniques, such as in [77], [119] and [120].

(2) The hierarchical approaches to limit the number of feedback messages by accumulating and filtering in subgroups, such as in [49], [97], [125], [126], [158] and [170].

Nevertheless, all of these solutions still have some disadvantages. Hierarchical approaches require the costly setup of the hierarchy of subgroups, and cannot be deployed in some scenarios, where there is a very limited capacity back channel (such as, satellite transmission with unicast backward channels). The feedback suppression by random timer and back-off techniques can reduce the feedback and recovery traffic, but cause high latencies for feedback and recovery.

### 2.6.2 Forward Error Correction (FEC)

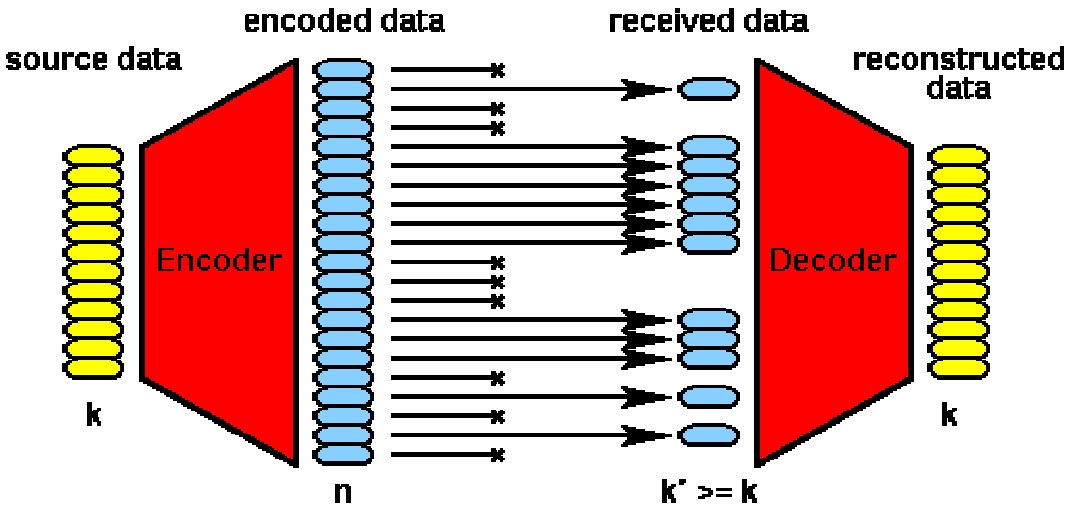



**Figure 2-10: Forward Error Correction** *(Source: [143])*

Forward Error Correction (FEC) (also known as *Parity Encoding*) refers to the ability to overcome both *erasures* (losses) and bit-level corruption [106]. Error control using FEC is based on the end-to-end recovery from the sender using FEC packets.

By using an erasure code (for example, based on the Reed-Solamon code structure - the theory of error correcting codes of [99]), the parity packets (also called *FEC packets, or Metadata*™) are encoded from the original data packets at the sender. For a group of *k* original packets that form a transmission group (*TG*), *n* different FEC packets can be encoded. Then, the sender strategically sends redundant data of *n* packets. After receiving the packets, the receivers decode the FEC packets to reconstruct the original data packets. The reception of any *k'* out of those *n* packets is sufficient to reconstruct the *k* original packets. If some packets are lost but more than or equal to *k* packets were received, there is no need to request repair packets. This would avoid the feedback and recovery implosion problem. If the received packets are fewer than *k* packets, the FEC packets will be retransmitted instead of the original

data packets. This improves the transmission efficiency, since a single parity packet (FEC packet) can repair the loss of any original data packets. In addition, different data packets lost by different receivers can be repaired with the same FEC packet [119].

There are two ways of using FEC: *proactive FEC* and *reactive FEC*. If the sender transmits extra FEC packets immediately following the original data of each *TG*, this is called proactive FEC. If the sender still transmits only *k* original data, but will use FEC packets instead of original data packet for recovery retransmission, this is called reactive FEC [98].

FEC seems to have a potential to reduce network bandwidth usage as reducing the end-to-end loss rate. Yet, FEC at the same time causes extra delay (to encode and decode FEC packets), and the implementation complexity. The performance seems to depend crucially on packet loss characteristics. The amount of redundancy added to the data stream would be a function of the loss rate of the receivers and the amount of the additional network bandwidth available [152]. This is still an open issue for the sender to decide the right balance.

So far, there are several FEC algorithms proposed, such as:

- Effective erasure codes of Rizzo based on Vandermonde matrices [145],
- *Tornado* codes [103],
- *Luby Transform (LT)* codes [101], used in *Digital Fountain™ (DF)* [32] technology.

### 2.6.3 ARQ vs. FEC

A comparison between FEC and ARQ is summarised in Table 2-4.

| Criteria | ARQ | FEC |
|---|---|---|
| Methodology | Error detection and Retransmission | Strategically sends redundant data with parity from the theory of error correcting codes. |
| Back channel | Requires back channel for feedback | Requires no back channel |
| Scalability | Good for Unicast<br>Not scalable to large number of receivers | Good for both Unicast and Multicast<br>Scalable to large number of receivers |
| Problems | Feedback Implosion<br>Unsuitability to a very limited capacity back channel, such as satellite transmission | Extra delay to encode and decode<br>Implementation complexity |

**Table 2-4: ARQ vs. FEC**

## 2.7 Summary

This chapter has provided related background of the issues covered in this thesis. The introductory materials of three modes of communication (unicast, broadcast, and multicast), IP multicast model, network congestion, fairness of bandwidth share, two directions of multicast congestion control, and error control of multicast protocols have been given. The hindrances in deploying multicast have also been discussed.

# Chapter 3

# Research Methodology

## 3.1 Introduction

This research aims at evaluating and proposing a new design of MR-MCC protocol. To do so, we have to establish a strong performance evaluation technique as our research methodology. This chapter explains how we choose and establish the research methodology. Following this section, we describe three choices of performance evaluation techniques (analytical modelling, simulation and measurement) in Section 3.2. Section 3.3 explains the significance of network simulation and the reasons for choosing it as a technique to evaluate MR-MCC protocols. Alternatives to network simulation packages and the reasons why we choose ns-2 are discussed in Section 3.4. In Section 3.5, we explain the steps of our simulation construction. Section 3.6 justifies simulation models used in this research. Section 3.7 reviews simulation parameter settings. Section 3.8 illustrates how we run

simulation and post-process the output data. Section 3.9 discusses results analysis and confidence intervals. Details of validation and verification are provided in Section 3.10. At the end, the summary of this chapter is given in Section 3.11.

## 3.2 Performance Evaluation Techniques

According to Law and Kelton [92], the real-world facility or process that we want to study/evaluate is called a *system*. In order to study it scientifically, a set of assumptions (in the form of mathematical or logical relationships) about how it works may be made. These assumptions are used to construct a *model* to understand the system. In our research context, the system referred to is network protocols, particularly MR-MCC protocols. To evaluate system performance, there are three possible techniques, namely analytical modelling, simulation, and measurement, as mentioned in [80].

Measurement can be done in a test-bed network or an operational network. This technique requires real equipment, code and time to run experiments. Monitoring is the key to this technique. It is the most realistic form of performance evaluation technique. Prototyping may be needed before being able to do measurement if the system is new or not available (such as a new network protocol). This technique is frequently beyond the reach of network researchers for the following reasons:

- It may be too disruptive to do a measurement by testing on real operational networks [92].
- It can be very expensive and requires accumulated experience to do the measurement, in particular if we have to prototype and build a test-bed network on an interesting scale.

- Test-beds can be difficult to reconfigure and share among researchers [23].

Hence, published network protocol work relies mainly on analytical modelling and simulation. Not everyone chooses to progress through to the measurement and prototyping phase.

Analytical modelling is a construction of a mathematical model of the system such as queuing networks and Petri-Nets. It is the cheapest and least time-consuming technique compared with the other two techniques. Results from analytical modelling can have better predictive values than measurement or simulation. According to [92], it is also a good tool to study overall characterisation. Yet, analytical modelling is not a good tool to study detailed behaviour. Keshav [87] identifies two main drawbacks of analytical modelling as follows: (1) it generally requires too many simplifications and assumptions (which may be inaccurate) about the real network; (2) it ignores interactions that can prove to be critical in practice. Most network protocols and systems are too complex to be realistically modelled using analytical modelling [92]. In particular, the sequence of events leading to network congestion can be complex and generally hard to analyse analytically [87].

Due to the drawbacks of analytical modelling and measurement, a simulation study is therefore useful and even necessary to study MR-MCC protocols. An implementation of a protocol in a simulator would bring out practical difficulties (that are sometimes hidden in a formal approach), and motivates new approaches [87]. In addition, network simulators can also be easily shared among researchers.

Simulation involves constructing a model for the behaviour of a system. To use simulation, the researchers have to decide what to simulate and at what level of detail, and drive it with an appropriate abstraction of the workload [80]. Generally, simulation generates raw data that must be interpreted using statistical tools.

In terms of accuracy, Jain [80] comments on this aspect of these three techniques (analytical modelling, simulation and measurement) as follows:

- The accuracy of analytical modelling is the lowest among the three techniques.
- Simulation can incorporate more details and require fewer assumptions than analytical modelling. Hence, it is often closer to reality.
- The accuracy of measurement can vary from very high to none. Although measurement sounds like the *real thing*, it may not give accurate results. This is because several environmental parameters (such as system configuration, type of workload, and time of the measurement) may be unique in the experiment. Hence, they may not represent the range of variables found in the real world.

We can summarise the comparison of the three performance evaluation techniques as shown in Table 3-1.

| Criteria | Analytical Modelling | Simulation | Measurement |
|---|---|---|---|
| Time Consumption | Small | Medium | High |
| Cost | Small | Medium | High |
| Accuracy | Low | Moderate | Various |

**Table 3-1: Comparison of performance evaluation techniques**

## 3.3 Network Simulation

Like other scientific areas, the study of telecommunication networks has accepted computer simulation as one of the most commonly used paradigms [127]. Network simulation has been the main research methodology used by several researchers working on the core of Internet development. A great deal of credible published work (such as in ACM SIGCOMM, IEEE INFOCOM, IEEE ICON, IEEE/ACM Transactions of Networking) has been done using network simulation.

According to Floyd [60], some research on proposed changes to the Internet can be done with analytical modelling or experiments in test-beds; yet, network simulation is an essential tool to explore proposals in environments that have not been realised in the current Internet but may be in the future. Moreover, the use of network simulation has brought substantial benefits to network research, such as improving validation of the existing protocols, providing rich protocol development infrastructure, giving an opportunity to study large-scale protocol interaction, as well as easing comparison of results across research efforts [30]. Simulations have played a vital role in characterising both behaviour of the current Internet and the possible effects of proposed changes to its operation [63].

So, in this research, we elect network simulation as our performance evaluation technique. In particular, we aim at evaluating the design of future multicast congestion control protocols, which are not readily available and would be expensive to prototype and build a test-bed for. Moreover, as mentioned in [92], such network protocols are generally too complex and dynamic to be investigated accurately through analytical modelling techniques.

## 3.4 Network Simulation Packages

We elect to use network simulation techniques to experiment with alternative MR-MCC algorithms under a wide variety of scenarios. Hence, the network simulation package is a very important tool for our research.

There are several network simulation packages available. Some packages (such as Network Simulator version 2 (ns-2) [121], REalistic And Large (REAL) [85], Yet Another Tiny Simulator (YATS) [25], and Global Mobile Simulator (GloMoSim) [65]) are available as free open-sourced software. On the other hand, some packages (such as OpNet™ [3] and ComNet™ [18]) are commercial simulation packages. We are interested in network simulation packages that are freely available and open-sourced only. Apart from the budget reason, this is because we need to have full control over our simulation package. In particular, we have to implement our own MR-MCC algorithms into the network simulation packages. This means the modification and expansion of the network simulator's source codes are compulsory.

### 3.4.1 REAL

REAL (Realistic And Large) [85] was written by Keshav in 1988, comprising of more than 30 network modules in the C language. Network topologies and simulation scenarios can be described in REAL using a simple scripting language called *Net Language*. REAL was widely used by the network research community until ns-2 had been developed and widely deployed.

### 3.4.2 YATS

YATS (Yet Another Tiny Simulator) [25] is a network simulator from Dresden University of Technology. During early parts of this work, with use of YATS we have done a few performance studies of TCP over ATM (Asynchronous Transfer Mode). Some of our publications using YATS are ([137], [138], and [139]). YATS is written in C++ and has a parser to read configuration scripts. The main purpose of YATS is to simulate ATM Networks. It also provides all the important TCP/IP mechanisms, such as Slow-start, Congestion Avoidance, Fast Retransmit, Fast Recovery, and Nagle's algorithm. However, it has a very limited facility to support multicast mode. So, the simulator is inappropriate for the tasks of our thesis.

### 3.4.3 GloMoSim

GloMoSim (Global Mobile Simulator) [65], from University of California Los Angeles (UCLA), is built on a scalable simulation environment for wireless and wired network systems. It is designed to use the parallel discrete-event simulation capability. GloMoSim currently supports protocols for wireless networks only. Hence, it does not suit our tasks. However, as stated on the UCLA website, GloMoSim should be extended to simulate a wired as well as a hybrid network with both wired and wireless capabilities in the near future.

### 3.4.4 Ns-2

Ns (Network Simulator) [121] is an object-oriented simulator developed as a variant of REAL in 1990. Ns-1 was made publicly available in 1995, and ns-2 was released in 1997. It is a part of the Virtual InterNet Test-bed (VINT) project at the Lawrence Berkeley National Laboratory (LBNL) [30]. So far, ns-2 has become the most widely used network simulation package. As it is used by many network researchers, support of ns-2 is relatively easy to find. Any problems with it can be sent to *ns-2 mailing list,* and generally would get very good responses from the ns-2 community. Ns-2 is also provided with modules to support multicast mode. As a result, we have chosen it as our network simulation package for this research.

Since it was released in 1997, ns-2 development has been progressing. The current release at the time of writing this thesis (August 2003) is version 2.27. However, the ns-2 version used in this research is *version 2.1b6a*, which was the most updated version by the time that we started our implementations (November 1999). This version is also the most widely deployed ns-2 version for multicast congestion control research, as evidenced by [34], [94], [95], [147], [166] and [169].

The ns-2 simulator is a discrete event simulator, in which the state variables change instantaneously at separated points in time. The discrete event nature of the simulator requires ns-2 to register events before the event's time of execution. From Figure 3-1, the full sequence of events that will occur in the simulation can be viewed as a linked list ordered in time of occurrence (called *event scheduler*). Registering a new event involves inserting the event into a linked list at the correct position.

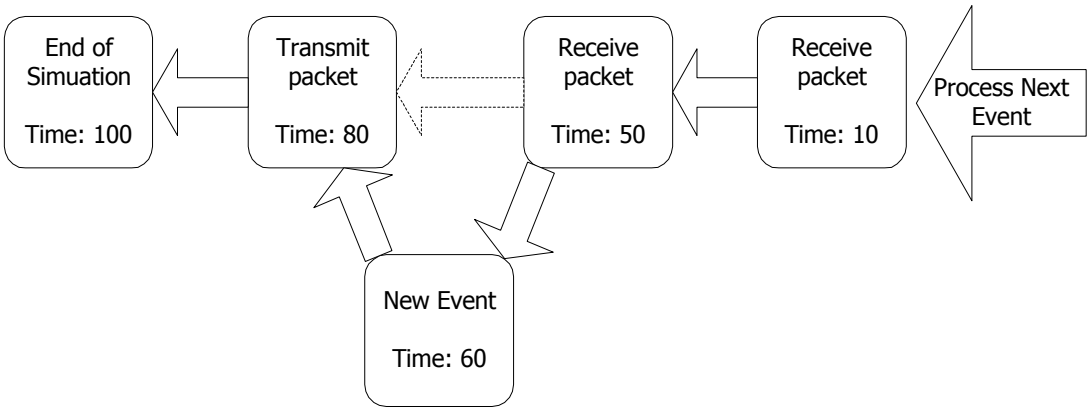



**Figure 3-1: Discrete event simulator**

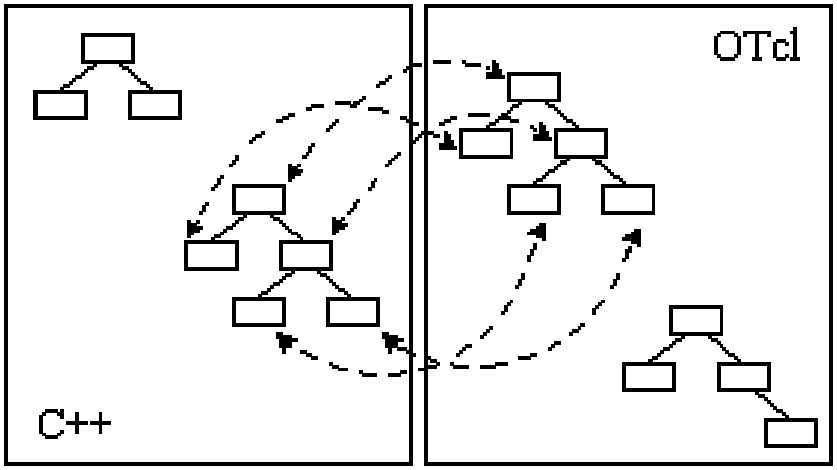



**Figure 3-2: C++ and oTcl** ***(Source: [40])***

As shown in Figure 3-2, ns-2 is written in both C++ and the *object-oriented Tool Command Language (oTcl)* [122] script language. Its core modules are implemented in C++ while its interfaces are implemented in oTcl. This is a powerful framework allowing the user to implement both network topologies and additional functionality through methods and procedures in oTcl scripts. This also eases the implementation of new modules in ns-2 by using oTcl and calling on the features or methods already implemented in the simulation core rather than a full C++ implementation. The compiled objects, variables and methods of ns-2 are made available for both C++ and oTcl interface via *oTcl Linkage*. The oTcl linkage creates a matching oTcl object for each C++ object, and makes the methods and the variables specified by the C++

object act as methods and variables of the corresponding oTcl object [40]. This oTcl linkage makes the learning curve for ns-2 quite steep to include oTcl both on the user-side and as function calls in the C++ code. However, once overcome, it provides a flexible and powerful simulation environment.

In order to simulate previous MR-MCC protocols (FLID-DL and PLM in Chapter 4), we use ns-2 version 2.1b6a, together with FLID-DL and PLM extended modules, provided by the authors of each protocol. Special thanks go to Luby, Goyal and Legout for fruitful discussion during the time we built and validated FLID-DL and PLM modules for our experiments.

In order to model, simulate and study our new design of an experimental MR-MCC protocol, we have implemented it in ns-2 version b6a. The details of its design and implementation are given in Chapter 5. Also, its simulation study is presented in Chapter 6.

## 3.5 Simulation Construction

Figure 3-3 illustrates steps of simulation construction of our research (in Chapters 4 and 6). These steps can be explained as follows:

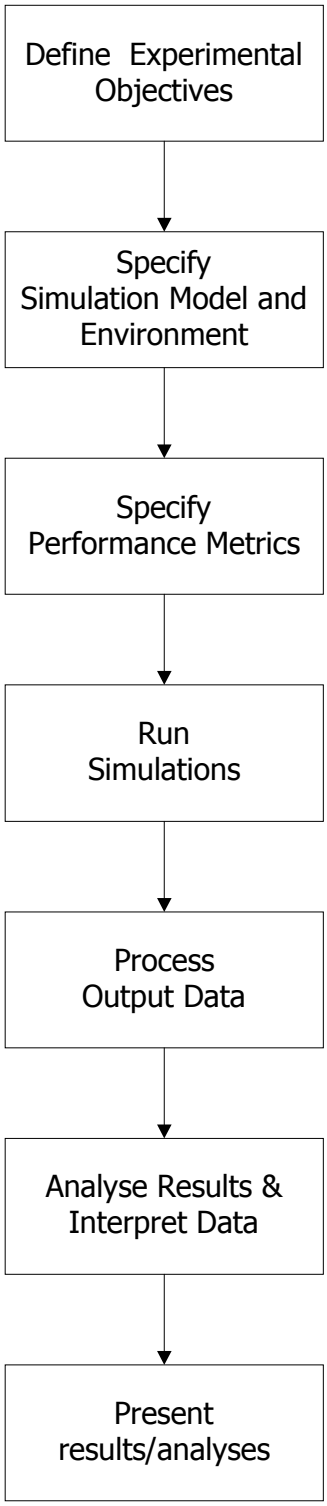



**Figure 3-3: Simulation construction**

**Define Experimental Objectives:** This is the step in which the objectives of the research are mapped into a set of simulation experiments. The system to be evaluated, goals, objectives and evaluation criteria are stated clearly in this step.

**Define Simulation Models and Environment:** This step provides an analysis of the problems to be tackled and specifies a simulation model. The simulation scenario, topology, network environment (traffic characteristics, simulation parameters including parameter sensitivity) are described.

**Define Performance Metrics:** *Performance Metric*, as explained in [80], refers to *"the criterion used to quantify the performance of the system"*. In this step, the performance metrics of the simulation are described in detail. In this thesis, several performance metrics (such as throughput, network utilisation efficiency, response time, TCP friendliness ratio, equality index, etc.) are used. Further detailed description of them will be given in Chapters 4 and 6.

**Simulation Phase:** In this step, simulation model, performance metrics and environment (scenarios, topology and configuration) are put together using network simulator input scripts and run. For our experiments in this research (Chapters 4 and 6), each simulation is run 20 times using a different *Random Number Generator (RNG)* seeds to get the averaged results, quoted with error bars with respect to confidence intervals of 95%.

**Post Simulation Process:** In this step, we collect the output data from the simulation phase to be analysed. This is done by a set of post-processing UNIX and EXCEL Visual Basic scripts.

**Results Analysis and Data Interpretation:** This phase undertakes analysis of the results obtained from simulation. It discusses, evaluates and interprets the results obtained.

**Presentation of Results:** This phase involves presenting and illustrating the results from simulation. The eventual goal of performance evaluation is to present the results and give the insight of the system. So, this phase is very crucial.

## 3.6 Simulation Model

The main parts of simulation models/scenarios used in this research are adjusted from the scenarios proposed by Handley et al. [109] as reference simulation models for testing multicast congestion control (presented in Reliable Multicast Group (RMT) meetings of IETF). Some are adjusted from the previous published work of MR-MCC protocols such as [34] and [95]. Also, some are modified from the reliable multicast simulation models of Digital Fountain [33].

We choose to use rather simple scenarios/models because of the following reasons:

- Time restriction: According to Jain [80], a majority of day-to-day performance problems in the real world are solved by simple models, due to time restrictions. He also comments as follows:

  *"Some analysts start with complex models that cannot be solved or a simulation project with very ambitious goals that are never achieved. It is better to start with simple models or experiments, get some results or insights, and then introduce the complication."*
- Understandability: Even if the time required to develop the model is not restricted, complex models are not easily understood. The influences of various parameters are mixed, thus confounding the results. They are therefore difficult to analyse correctly.

We have found that the simple scenarios used in our experiments are useful to explore behaviours of MR-MCC mechanisms in a well-characterised environment, and provide insights into the MR-MCC performance, as well as serve their objectives. Each scenario and parameter has been carefully selected, and parameter sensitivity

has been tested cautiously. However, we do not claim that the scenarios used in our experiment are a universal benchmark, and it is not our intention to create one in this work. We do, however, recommend that researchers consider our scenarios among others.

According to Floyd et al. [63], it can be very cryptic to evaluate the results from a single simulation or set of simulations. They recommend researchers to take great care in interpreting simulation results, and drawing conclusions from them. Furthermore, they suggest that researchers should reap sound insight and understanding from simulations, while never mistakenly taking simulation for the real world.

Although we do consider real life situations in defining our model, the scenarios used in our experiments are not intended to accurately model the Internet. We accept that our simulation model is rather simple, especially when compared with the real complex public Internet. Due to its heterogeneity, complexity and rapid changes, finding a suitable simulation model that simulates the Internet is fundamentally difficult, and is still an open research issue. A few studies (such as [36], [53], [63], [171], and [172]) have tried to tackle this issue. However, it is not within the scope of this research to simulate the Internet.

## 3.7 Simulation Parameter Settings

According to Jain [80], parameters are referred to as system characteristics that affect the performance of the system. However, not all parameters have an equal effect on the performance. It is important to identify those parameters which, if varied, will make a significant impact on the performance. The final outcome of simulation depends heavily on the set of values chosen for each parameter. Hence, in this section, we describe some important parameter settings used in our simulation. In addition, we have also performed *Parameter Sensitivity Analysis* to understand whether the conclusion would change if the simulation were run in slightly different parameter settings.

### 3.7.1 Packet Size

The size of packets in real life can be various and negotiated. The maximum packet size for Fibre Distributed Data Interface (FDDI) is 4500 bytes [83], while that for Ethernet LAN is 1500 bytes. RFC-2225 [91] suggests that the packet size for TCP over ATM network would be 9180 bytes. The default packet size for TCP/IP is 576 bytes [135], although many implementations round this down to 512 bytes [157].

Floyd et al. [62] point out that close to 100% of the packets in the Internet are 1500 bytes or smaller. According to [41], [42] and [64], the real life *typical* packet size distribution is *Tri-Model*, with the modes at or around 40 bytes (the size of ACK), 576 bytes (default of TCP/IP), and 1500 bytes (typical packet size for Ethernet).

We use the same packet size at 512 bytes for all protocols (TCP, PLM, FLID-DL, WEBRC and ERA) tested in our experiments. This is to ensure that the data packets

from each protocol are treated fairly by the routers with regard to the size of the packets.

For each experiment, we also vary the packet size from 512 bytes to 1024 and 1500 bytes to test the parameter sensitivity towards the size of packets. In all experiments, we have found that the bigger packet sizes consistently give the same trend of results.

### 3.7.2 Traffic Direction

Like most of the simulation work in this field, we only consider unidirectional traffic in our simulation model. On the forward path, the source sends the data packets to the receiver, and the receiver sends nothing except ACK packets back on the reverse path. This unidirectional traffic is generally used by the network simulation community to avoid the effect of *ACK Compression* [115], [167], [173].

According to [115] and [167], ACK compression is the phenomenon that could undermine the correctness of the throughput results. It can be explained as follows:

- If bi-directional traffic were used in any experiment, the data packets transmitted by the connections in one direction should share the same physical path with the ACK packets from the connections in the opposite direction.
- These ACK packets may be in the same queues with other data packets from the opposite direction during their transit to the source.
- Hence, by the queuing delay, the arrival interval of these ACK packets may be changed. A bunch of ACK packets can arrive closer to each other than they were sent, and the sender in this case could be misled to send more data than the network can accept, resulting in network congestion and packet loss.
- The results of ACK-compression are an unfairness of the throughput received at competing connections, and a reduction in the overall throughput [167].

In our experiments, we want to investigate how throughput would be affected from network congestion when using different congestion control mechanisms. Hence, using unidirectional traffic is useful to exclude the ACK compression effect from confounding our results.

### 3.7.3 Characteristics of TCP Used in Our Simulation

**Flavour of TCP**

For the experiments that also simulate the bandwidth share between TCP connections and MR-MCC connections, we use the ns-2 implementation of TCP Reno, which is the most commonly used flavour of TCP [128]. Actually, TCP with *Selective Acknowledgement (SACK)*, proposed in [111], has improved the loss recovery scheme, and its deployment is now increasing [20]. According to [19] and [20], the SACK fraction of TCP flavours around the world from December 1998 to February 2000 has increased from 8% to 40%. Most new operating systems (such as Win95/98/NT/2000/XP, Linux, Solaris, IRIX, and OpenBSD) have now provided TCP SACK. However, several studies (such as [34], [102] and [159]), which test the bandwidth share between MR-MCC protocols with TCP Reno or TCP SACK, show the same trend of results between Reno and SACK. Hence, we consider only TCP Reno as a representative of TCP protocol in our experiments.

**Maximum Size of TCP Congestion Window**

According to RFC-2581 [21], the standard congestion window size is 64 Kbytes. However, the maximum size of TCP congestion window in our experiments is set to 2000 packets to remove the effect of the maximum window size. So, the throughput of TCP will not be limited by the congestion window size but will rely on its congestion control mechanism. This technique is also used in many network simulation publications, such as [34], [68], [95], [138] and [139].

### 3.7.4 Router Characteristics

According to [68], a router queue has two elements, *Buffering* and *Scheduling* as shown in Figure 3-4.

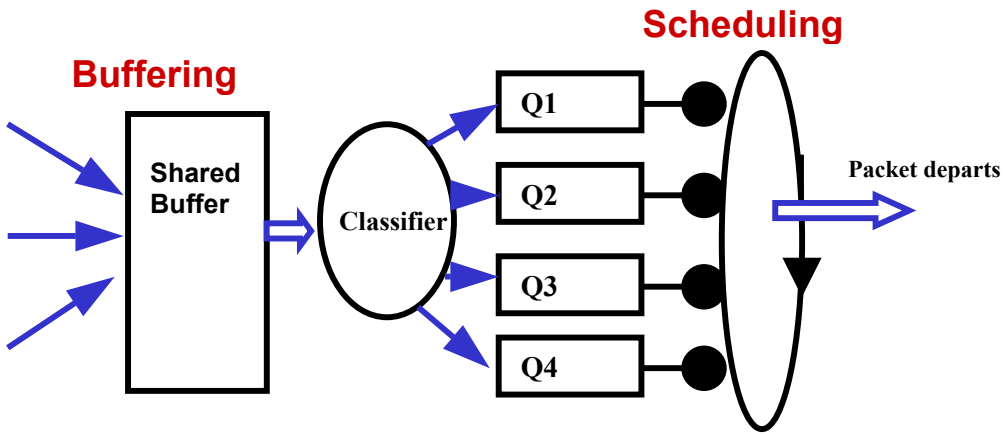



**Figure 3-4: Functional elements in router queue**

(1) **Buffering:** Buffer management is performed by routers to control the number of packets entering the router. Buffering is used to absorb *traffic bursts*. The commonly used buffer management scheme is *Drop-tail*. There are also other sophisticated buffer management schemes proposed, such as *Random Early Detection (RED)* [61], and *Random Early Marking (REM)* [100].

(2) **Scheduling (also called Queuing):** The scheduling determines how packets are scheduled onto the next hop. The most commonly used scheme of scheduling is a *First-In-First-Out (FIFO)* queue. There are also other sophisticated scheduling schemes proposed, such as *Round Robin Queue, Fair Queuing (FQ) and Weighted Fair Queue (WFQ)*.

The type of router used in our experiments (in Chapters 4 and 6) is drop-tail FIFO, which is the most commonly used, and supported by every router. Yet, for the

experiments of PLM (in Chapter 4), the router's queuing scheme is FQ, as required by the PLM specification.

The router buffer is used to absorb bursts of traffic. On one hand, a big enough buffer size is needed to absorb the bursts. On the other hand, to minimise the queuing delay, it is required that the average buffer size should not be too high. Different traffic types prefer different sizes of buffer size. For the *Elephant Traffic* (bandwidth sensitive traffic), a big average buffer size is preferred in order to absorb bursts. Conversely, for the *Mouse Traffic* (delay sensitive traffic), a small average buffer size is preferred, since it wants to keep the queuing delay low. In real life, the router's buffer size can be various, and be tuned by network administrators.

In our experiments, the buffer size of routers is set to twice delay-bandwidth product. The delay-bandwidth product is *a number of packets on flight* or the amount of data that can be moving on a network at any given time [76]. The purpose of setting it like this is to provide enough buffer size to absorb the burst at two times the number of packets on flight. In general, the common buffer size used in network simulation published work (such as [24], [34]) varies from one to three times delay-bandwidth product. The buffer size below one delay-bandwidth product is used only when the experimental simulation has a special purpose to test the system in the situation of having scarce buffer resource (which is not our purpose).

We also test the parameter sensitivity of buffer size by varying it from one to four times delay-bandwidth product. The results still show the same trend. However, when

we try our experiments at scarce buffer size below one delay-bandwidth product, we find that performance decays.

## 3.8 Simulation and Post-Simulation Phases

This section discusses how we run our simulation and post-process the output data. The problems found during running the simulation and our solutions to deal with them are also described.

Pawlikowski et al. [127] have revealed a crisis of mistaken analyses of network simulation results. According to their survey, more than 70% of simulation-based publications on telecommunication network were not concerned about the random nature of output data obtained from stochastic simulation studies, and either did not care to mention that final results were outcomes of appropriate statistical analyses, or even reported purely random results. Moreover, they suggest that a credible simulation study must include two important practices: (1) use of an appropriate *RNG*, and (2) suitable statistical analyses of simulation output data. Consequently, we run each simulation at least 20 times to gain the average results and quote it with error bars with respect to confidence intervals of 95% (further details explained in Section 3.9).

However, in our experiments, we have several scenarios to be simulated, and each scenario has several cases of parameter variation. So, to run each simulation case 20 times can consume a very long period of simulation time. To tackle this problem, we therefore write Unix and oTcl scripts to distribute the simulations to be run in parallel over more than 140 Linux machines of the School of Computing, University of Leeds.

This helps cut down on simulation time tremendously. Our *Simulation Control Scripts* can also specify the time that the simulation would be started so as not to disturb system during peak hours. Furthermore, they command several simulations to be run on the machine continuously, i.e., after finishing a simulation (approximately 3-4 hours), the next simulation is scheduled to be run on that machine. By using these scripts, we are therefore able to use machines in the School efficiently, and run each simulation 20 times within a decent period of time.

On completion of the simulation, results from all the output files are extracted using other *Post-simulation Unix Scripts*. After that, an Excel Visual Basic script reads these summarised results into a spreadsheet. The script also plots all these results in the format wanted. Our simulation oTcl scripts and examples of our simulation control scripts as well as other related scripts are available online at the author's *Research Log Web Page* [136].

## 3.9 Results Analysis and Confidence Intervals

According to Jain [80], if a simulation or measurement were repeated several times, the results would be slightly different each time. Simply comparing the average result does not lead to correct conclusions, in particular if the variability of the result is high. So, in this thesis, our results are averaged and quoted from 20 runs with respect to confidence intervals of 95%. The error bar is shown where it is appropriate. We do not show error bars when intervals are very small or negligible.

Our averaged results (also called *Mean* or *Expected Value*) and their *confidence intervals* have been obtained by the following steps:

- After repeating a simulation with different RNG for $n$ runs, we can calculate an averaged result ($\overline{X}$) as:

$$\overline{X} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{3-1}$$

where $x_i$ = the result of each run, and $n$ is the number of runs.

According to Jain [80], this averaged result should not be given as *an absolute certainty*, but it provides information with *a probabilistic bound* only. We can find two bounds (for example, $b_1$ and $b_2$), such that there is a high probability, '$1-\alpha$', that the result is in intervals $(b_1, b_2)$.

- In statistics, the $(b_1, b_2)$ interval is called the *Confidence Interval* of $\overline{X}$, $\alpha$ is called the *Significance Level*, '$100(1-\alpha)$' is called the *Confidence Level*, and '$1-\alpha$' is called the *Confidence Coefficient*.

- In our case, we use a 95% confidence level. So, the averaged results with their confidence intervals of our results can be quoted as $\bar{x} \pm t_{\alpha/2}\frac{s}{\sqrt{n}}$, where $t_{\alpha/2}$ is *t-value* (the *Interval Coefficient*) with an area of $\alpha/2$ to its right, $n$ = number of runs (which is 20), $\alpha$ is the significance level (which is 0.05 in this case), $s$ = *Standard Deviation* of $n$ runs (calculated from Eq. (3-2)).

$$s = \sqrt{\frac{n\sum_{i=1}^{n} x_i^{\,2} - (\sum_{i=1}^{n} x_i)^2}{n(n-1)}} \tag{3-2}$$

## 3.10 Validation and Verification

According to Jain [80], validation is a process to ensure that the assumptions used in developing the model are reasonable, while verification is a process to ensure the correctness of the implementation of the model. In this report, we use *validation* for both validation and verification of the definitions given by Jain.

### 3.10.1 Validation of Network Simulator

Ns-2 is provided with a large collection of detailed validation scripts. By using these scripts, it performs self-validation as a part of the build process. By this self-validation build process, the simulator is run using a specific set of input values with known outputs. Then, the outputs from the self-validation are compared with the known output to validate the results. Finally, the user is notified if ns-2 fails to validate any of its components during the build process. This self-validation process is used to validate the functionality of TCP Reno and other network objects used in our simulation experiments.

### 3.10.2 Validation of ERA Implementation on ns-2

We validate our ERA implementation on ns-2 by using *Run-time Trace* and *Incremental Implementation*. The details will be given in Chapter 5. Furthermore, we have made ERA modules available online at our *Research Log Web* [136] to be further validated by any interested researchers.

### 3.10.3 Validation of FLID-DL and PLM Modules

The ns-2 modules of PLM and FLID-DL are provided by the authors of each protocol. Both have been made publicly available and extensively tested by several research groups, and have formed a great deal of published work (such as [67] and [174]).

In particular, PLM modules have been included with the ns-2 package from version b7. The modules are provided with a set of validation test-suites. The test-suites simulate PLM using a specific set of input values with known outputs. Then, the outputs from the test-suite are compared with the known output to validate the results. We use this test-suite to validate PLM used in our experiments.

FLID-DL modules have been provided with many samples of simulation scripts and results that we can rerun to test their validity. Moreover, to ensure the validation of FLID-DL modules, we have modelled a validation scenario to test them. The details are given in Appendix A.

### 3.10.4 Further Validation

Although we have made every effort to validate our simulation experiments, the validation of scientific work is never completely finished. The scientific experiments and theories could be re-assessed again and again as the time passes. According to Day [46], good scientific work should be made available and disclose enough information to enable peers to assess observation, repeat experiments, and re-evaluate. Hence, the ERA ns-2 modules and the simulation scripts used in this research and other relevant information have been made available online at our *Research Log Web*

*Page* [136] to be further validated. Furthermore, we are grateful to ns-2 programmers/users, who have already fed back on this page.

## 3.11 Summary

In summary, we have described three alternatives (analytical modelling, simulation, and measurement) of performance evaluation techniques and their pros and cons. Our research goal is to evaluate previous MR-MCC protocols, and propose a new design of future ones. To reach the goals, we have elected to use network simulation as our technique. The reasons are: (1) it would be too expensive and difficult to prototype, build a test-bed, and take measurement as the technique to reach our goal; (2) analytical modelling is also not a good tool to investigate accurately the detailed behaviour of such a network protocol, which is generally too complex and dynamic.

The ns-2 network simulation package then has been chosen for its robustness and wide support. In addition, we have described how we construct our simulation, including defining simulation objectives, setting up simulation model and parameters, running simulation, processing output, and analysing, as well as presenting the results. The validation and verification of the network simulation package and relevant extended modules for ns-2 (FLID-DL, PLM, and our ERA) are also discussed.

# Chapter 4

# Performance Study of Some Earlier MR-MCC Protocols

## 4.1 Introduction

As mentioned in Chapter 2, the lack of proper multicast congestion control mechanisms is one of the main obstacles to the deployment of multicast over the Internet. To achieve scalability in a very large heterogeneous group of receivers over the Internet, MR-MCC has been accepted by the research community as a solution for multicast congestion control. In this chapter, we focus on investigation and performance study of some significant proposals of MR-MCC protocols – RLM, RLC, FLID-DL and PLM.

FLID-DL [34] and PLM [95] are two recently proposed MR-MCC protocols. Both protocols have been extended from the famous archetype MR-MCC protocols (RLM [113] and RLC [159]) by improving the efficiency of network utilisation, protocol fairness and fast convergence. Both have been evaluated by their authors and claim a few advances for layered multicast congestion control protocols over a broad range of conditions. However, the authors of both protocols have conducted performance evaluation with respect to their own work and the archetype protocols only. To the best of our knowledge, there is no known study that evaluates these protocols in comparison to each other.

So, in this chapter, we conduct a performance evaluation to compare them by using network simulation. Our main goal is to explore both protocols under certain network conditions to understand the advantages and disadvantages of them comparatively. We hope this may lead to proposals for further enhancement to MR-MCC.

To do so, we start from reviewing the proposal of RLM, RLC, FLID-DL and PLM. Then, we have integrated PLM and FLID-DL modules provided by the authors of each protocol into the network simulator ns-2 [121]. We then specify a set of criteria to evaluate MR-MCC protocols, such as responsiveness, the efficiency of network utilisation, fairness, TCP-friendliness, packet loss ratio, scalability, fast convergence, smoothness and feasibility. After that, we have modelled, designed and run several simulation experiments to evaluate them. The simulation results demonstrate that PLM outperforms FLID-DL in terms of TCP-friendliness, smoothness, packet loss, fast convergence and efficiency.

The remainder of this chapter is organised as follows. We begin in Sections 4.2 and 4.3 by discussing some details about RLM and RLC, the archetype protocols that both FLID-DL and PLM are based on. Then, FLID-DL and PLM are discussed consecutively in Sections 4.4 and 4.5. In Section 4.6, we theoretically compare PLM with FLID-DL. In Section 4.7, we propose a set of criteria to evaluate MR-MCC protocols. Simulation models, tools, scenarios and performance metrics for the experiments are presented in Section 4.8. Section 4.9 discusses performance results. In Section 4.10, a summary of this chapter is given.

## 4.2 RLM

The Receiver-driven Layered Multicast (RLM) protocol (proposed by McCanne et al. [113]) is the first proposal of multicast congestion control using layered coding and receiver-driven approaches. It has been proposed for packet video transmission to large and heterogeneous audiences. RLM uses the technique called *Join Experiment* to adjust receivers' reception rate to the network condition. This involves the receiver increasing its reception rate by periodically subscribing to an additional layer and decreasing its reception rate by unsubscribing from a layer when experiencing packet loss. There are several fundamental problems of RLM reported in the literature and summarised as follows:

1. Uncoordinated join experiments by downstream receivers create substantial problems. A receiver's join experiments can introduce packet losses at other receivers behind the same bottleneck link. This finally results in unfairness among the downstream receivers (known as *intra-session unfairness*). To tackle this problem, the authors of RLM have proposed to use a synchronisation control message to coordinate the join experiment. Yet, this synchronisation control message could introduce a scalability problem [159].
2. Several studies (such as [88], [95]) have reported that RLM exhibits neither inter-protocol nor intra-protocol fairness.
3. Join experiments per se are prone to packet losses when oversubscribed causing bandwidth waste.
4. When receivers try to tackle congestion by unsubscribing from a layer, it can take several seconds to take effect, due to *the IGMP Leave Latency Problem* (mentioned in Chapter 2). This can slow down the bandwidth recovery, and therefore causes congestion persistence.

5. As reported in [94], RLM has a very slow convergence adaptation scheme to an optimal rate. It can take several minutes to do a join experiment to discover available bandwidth.
6. Finally, RLM has no support for error recovery. Hence, it cannot support reliable multicast applications and is proposed only for multimedia applications.

## 4.3 RLC

| Criteria | RLC | RLM |
|---|---|---|
| Layer granularity | Base rate = $R_o$<br>Cumulative Rate of layer *i,*<br>$R_i = 2^i * R_o$ | Depend on CODEC |
| Error recovery scheme | Erasure code FEC Encoding | None |
| Supported applications | Both multimedia and reliable multicast applications | Only multimedia applications |
| Bandwidth probe | Synchronised Join Experiment and Burst Test | Un-coordinated Join Experiment |
| Coordination of receivers | SPs | Synchronisation control message |
| Fairness | Claims to be fairer than RLM | No Intra-session fairness<br>No Inter-protocol fairness<br>No Intra-protocol fairness<br>No TCP-friendliness |

**Table 4-1: RLM vs. RLC**

Receiver-driven Layered congestion Control (RLC) was proposed by Vicisano et al. [159] to address some of the problems in RLM. First, RLC introduces FEC encoding for reliable multicast applications. This enables receiver-driven layered multicast to be applied not only to multimedia applications but also to reliable multicast applications like database replication or bulk data transfer. Second, RLC introduces *Synchronised Join Experiments* and *Burst Test* techniques to adapt reception rate to network conditions. Table 4-1 summarises the comparison between RLM and RLC.

*Synchronised Join Experiments* and *Burst Test* can be summarised as follows:

- The source places *Synchronisation Points (SP)* into outgoing data streams as increase signals. SPs are special packets in the data stream containing outgoing information about the number of layers and their respective data rate. An SP contains an increasing signal, which indicates the point at which receivers may join a specific layer.
- A receiver can only subscribe to a new layer after receiving an SP, and can subscribe to layer *i+1* when it receives an SP in layer *i*. The use of SPs deals very well with the problems faced by uncoordinated join experiments in RLM.
- After subscribing, if a receiver experiences packet losses, it will unsubscribe and drop back to the original layer before the join experiment.
- Due to the IGMP leave latency problem, to unsubscribe can be very slow and push the network into a state of congestion for a long-standing period. To tackle this problem, RLC uses a burst test technique, where the source periodically injects a brief burst of packets on each layer prior to a SP on that layer. This burst is to simulate the rate of the next layer before real subscription. After the burst test, if there were no packet loss, it would be safe to subscribe to the next layer.

RLC proposes the layer rate should use a doubling scheme, i.e., the rates through the cumulative layers are 1, 2, 4, 8 … Adding a layer will double the rate, and dropping a layer will halve the rate. So, RLC uses *Doubling Increase Doubling Decrease (DIDD)* rate adjustment. To achieve the same rate adjustment scheme as TCP, i.e., *Additive Increase Multiplicative Decrease (AIMD)* at a coarse grain, RLC places SPs on layer *i* at a frequency of $1/R_i$, where $R_i$ is the cumulative rate through layer *i*.

RLC still suffers from several problems. Its doubling scheme can cause dramatic fluctuations in network bandwidth consumption and rapid queue build-up [34]. After burst test and finding no loss, it still cannot be ensured that adding a layer will be safe. So, its join experiment even with the burst test cannot ensure avoiding packet losses and over-subscription. Due to the IGMP leave latency problem, to unsubscribe from a layer can be very slow. So, congestion can persist. In addition, Legout et al. have shown (in [94]) the pathological behaviour of RLC, where it becomes very unfair towards TCP and slow to converge in layer subscription. From these experiments, RLC has been revealed to be very sluggish before reaching the optimal layer, causing inefficiency of bandwidth usage. Furthermore, it is revealed in [88] that RLC is designed to be fair towards TCP with an RTT of one second only. The experiments in [88] have shown that RLC becomes aggressive in competing with TCP when RTT of TCP is larger than one second, and too conservative when RTT of TCP is much less than one second.

## 4.4 FLID-DL

Fair Layer Increase Decrease with Dynamic Layering (FLID-DL) is proposed by Byers et al. [34] to address some deficiencies of RLC. The protocol introduces two innovative techniques – *Dynamic Layering (DL)* and *Fair Layer Increase Decrease (FLID)*.

With the DL mechanism, the source partitions transmission time into *Time Slots* of a predefined duration *T* seconds each. Then, it generates a set of *Dynamic Layers* by decreasing transmission rate of each layer in a series of steps (time slots) until reaching a zero rate. Then, the rate remains at this zero state for a certain period of time called a *Quiescent Period*. In order to limit the total number of layers required by the mechanism, the same pattern of dynamic layers is reused after the quiescent period.

The receivers control their reception rate autonomously by subscribing to a certain number of dynamic layers. To maintain their reception rate, the receivers have to subscribe periodically to a certain number of additional layers. To increase their reception rates, they must subscribe to more additional layers. To reduce their reception rate (due to detected congestion), the receivers simply do not subscribe to additional layers. There is no need to unsubscribe since the transmission rate is automatically dropped over time. Hence, the DL mechanism helps mitigate the IGMP leave latency problem. It gracefully reduces the leave latencies associated with unsubscribing from a layer.

| Criteria | FLID-DL | RLC |
|---|---|---|
| Layering scheme | Multiplicative with any constant *Rate Multiplier* | Multiplicative scheme with doubling steps |
| Rate adaptation | Multiplicative Increase Multiplicative Decrease (MIMD) | Doubling Increase Doubling Decrease (DIMD) |
| Cumulative rate | Base Rate = $R_o$ Cumulative Rate of layer *i,* $R_i = C^i * R_o$, where *C* = Rate Multiplier | Base Rate = $R_o$ Cumulative Rate of layer *i,* $R_i = 2^i * R_o$ |
| Rate Increase | Milder rate increase | Abrupt rate increase |
| Bandwidth probe | Probabilistic Synchronised Join Experiment | Deterministic Synchronised Join Experiment + Burst Test |
| IGMP leave latency | DL to ameliorate the problem | N/A |
| Packet loss | Less | More |
| Convergence | Faster | Slower |

**Table 4-2: FLID-DL vs. RLC**

However, this advantage does not come without cost. To avoid unsubscribing processes, multiple subscribing processes are used instead for the dynamic layering scheme. This causes excessive control messages produced by FLID-DL.

The FLID mechanism is used to tackle the abrupt rate increase in RLC by choosing to increase the rate with a milder step. As shown in Table 4-2, instead of having the fixed *Rate Multiplier* equal to two like RLC, FLID generalises the layering scheme by having the rate multiplier equals to any predefined constant factor. With the recommended rate multiplier of 1.3 [34], the rate increase of FLID-DL is milder than RLC. Hence, the possibility of packet loss is reduced.

The FLID mechanism maintains RLC's style of SPs to coordinate receivers, and still uses the join experiment concept. However, FLID does not use the *Burst Test* technique, but uses probabilistic increase signals instead. Receivers subscribe to additional layers only with a certain probability. This probability and the scheme of spacing out the increase signals placed into packets (i.e., SPs) are chosen in such a way that, on average, the reception rate increase by receivers without packet loss is linear, i.e. achieving a compatible rate with TCP. The rate adaptation at the receivers can be summarised as follows:

```
For each time slot
   If packet loss is detected then
      Decrease subscription level by 1 at the end of the time slot
      If layer dropped is the lowest then
         EXIT the session
      End If
    Else if there is no packet loss detected then
      If the receiver receives an SP of the higher subscription level
      then
         It will increase its reception rate
         by increasing the subscription level by one
         at the beginning of the next time slot.
      Else
         The receiver maintains the current reception rate
      End If
   End If
Next
```

FLID-DL claims a few improvements over RLC as follows:

- mitigating the problem of IGMP leave latency by using DL,
- giving better fairness properties than RLC,
- giving better efficiency of network utilisation than RLC,
- causing less packet loss than RLC.

## 4.5 PLM

Packet-pair receiver-driven cumulative Layered Multicast (PLM) was proposed by Legout et al. [95] to improve RLC. It is based on the receiver-driven cumulative layering scheme. However, to react properly to congestion, PLM requires that all layers must follow the same multicast routing tree. The key mechanisms of PLM are *Receiver-side Packet-Pair Probe* (PP) and *Fair Queuing (FQ)* at routers.

Instead of relying on a join experiment technique like RLM, RLC and FLID-DL, PLM uses a PP approach to infer the available bandwidth, as well as to avoid congestion. With the PP approach, a PLM source periodically sends a pair of its data packets as a burst to infer the bandwidth share of the flow. It uses a one-bit field of the packet header to indicate the first packet of a pair. At the receiver side, the estimation of available bandwidth is calculated from packet size divided by the inter-arrival gap. The first PP that leaves the queue after congestion occurs will be a signal of congestion. The estimated bandwidth is used to adapt the rate. This is done only once at every regular *Check Period (C)* interval to avoid oscillatory rate adaptation.

With respect to the estimated bandwidth, PLM receivers use the convergence algorithm to utilise efficiently the available bandwidth and avoid congestion by adapting subscription rate as:

```
If  (estimated bandwidth < subscribed bandwidth) then
   Drop layers until (subscribed bandwidth < estimated bandwidth)
   If layer dropped is the lowest then
      EXIT the session
   End If
Else If  (estimated bandwidth ≥ subscribed bandwidth)
   Subscribe more layers
      while (subscribed bandwidth  < estimated bandwidth)
End If
```

To deal with the case when packet pairs are lost during severe congestion, PLM defines a *Timeout Period*. If no packet is received before the timeout, a layer will be dropped due to the expectation of congestion. If some packets are received but no PP is received, and the loss rate is over a predefined *Loss Threshold*, a layer will be dropped. Then, in order not to over-react to loss, PLM will wait for a predefined *Blind Period* before re-estimating the loss rate.

PLM assumes the deployment of a fair queuing mechanism in routers, and relies on a fair scheduler to ensure fairness, including intra-protocol fairness, inter-protocol fairness and TCP-friendliness.

PLM has a few advantages over RLC. First, it has a faster convergence for rate adaptation. It can quickly reach the optimal level of layer subscription. Furthermore, compared with join experiment, PP can sense the bandwidth changing in the network before congestion becomes severe (i.e., before packet loss). Finally, the fair scheduler makes PLM very intra-protocol and inter-protocol fair. However, it is still arguable whether the fair scheduler required in the router is feasible to be implemented over the whole Internet.

## 4.6 Comparison of PLM and FLID-DL

| Criteria | PLM | FLID-DL |
|---|---|---|
| Layering scheme | Any (unspecified) | Multiplicative |
| Bandwidth-Probing Mechanism | Receiver-side packet-pair probe | Probabilistic synchronised join experiment |
| Proposed Solution to IGMP leave latency problem | N/A | Use DL to mitigate this problem |
| Convergence | Claims to be faster than RLC | Claims to be faster than RLC |
| Packet loss | Claims to be less than RLC | Claims to be less than RLC |
| Fairness Issue | Claims to be better than RLC<br>Use Fair Queuing | Claims to be better than RLC<br>Use layer distribution matched with TCP equation |

**Table 4-3: PLM vs. FLID-DL**

Table 4-3 summarises PLM and FLID-DL in comparison. FLID-DL specifies to use a multiplicative layering scheme, while PLM leaves layering unspecified. Byers et al. [34] claim for FLID-DL that their multiplicative scheme with a milder rate increase step is a better choice than the doubling rate increase of RLC.

| Level | Rate | Quality of Service |
|---|---|---|
| 1 | 10 Kbps | GSM quality |
| 2 | 32 Kbps | LW radio quality |
| 3 | 64 Kbps | 2 stereos |
| 4 | 128 Kbps | FM radio quality |
| 5 | 256 Kbps | 4 stereos |

**Table 4-4: Quality levels of audio broadcast** (*Source: [95]*)

We argue that an unspecified layering scheme is even better. In particular, for multimedia applications, layer organisation depends highly on the CODEC used, which may not be easy or possible to adjust using a multiplicative (or any specific) scheme. Significantly, the perceived quality and the requirement of bandwidth are actually the key to layer organisation. Too fine-grained adjustment may be useless, as it cannot be perceived or cannot improve the user's satisfaction. For instance, in the case of audio broadcast shown in Table 4-4, there is no point in adjusting the rate from 10 to 20 Kbps, as this adjustment does not improve the user's satisfaction. However, the layering scheme would be correspondent to the perceived qualities as shown in the table. In this case, the PLM unspecific layering scheme is easier to set to the perceived qualities and the rate requirements than the multiplicative scheme of FLID-DL.

The DL mechanism of FLID-DL introduces a solution or at least mitigation to the problem of IGMP leave latency while PLM has no technique to tackle this problem. PLM leaves the multicasting model to rely on Fast IGMP proposed in [146] to tackle this problem instead.

For a bandwidth-probing mechanism, PLM uses receiver-side PP, while FLID-DL uses a probabilistic synchronised join experiment. By relying on their bandwidth-probing and rate adaptation mechanisms, both protocols claim to be more responsive, cause less packet loss, and exhibit faster convergence than RLC. In our simulation experiments, we set out to show which is better.

For the fairness issue, PLM relies on a fair queuing mechanism at routers, while FLID-DL tries to handle distributed rate through layers to be compatible with TCP. For PLM, the fair queuing assumption is arguable in terms of feasibility over the whole Internet. So, our experiments in Section 4.8.6 are designed to investigate the fairness property of PLM when there is no FQ at routers. For FLID-DL, its fairness mechanism does not take into account the RTT accurately. It has only a fixed simulated RTT value as a parameter, while the real RTT to different receivers can be different. As a result, we expect FLID-DL to exhibit unfair behaviour towards TCP under certain network conditions.

## 4.7 Evaluation Criteria of MR-MCC Protocols

In this section, we establish a set of criteria to evaluate MR-MCC protocols as follows:

### 4.7.1 Responsiveness

The first goal of congestion control protocols is to be responsive to congestion. A good protocol should be able to detect congestion signals quickly and reduce its transmission rate before causing congestion collapse. For MR-MCC protocols, responsiveness depends mainly on how receivers detect congestion and how quickly they react to it by dropping layers. In particular, the IGMP leave latency problem (mentioned in Chapter 2) is a hindrance to the responsiveness of MR-MCC protocols.

### 4.7.2 High Network Utilisation

MR-MCC protocols should be able to achieve a high network utilisation. When the network bandwidth becomes available, a good protocol should not leave it under-utilised. This depends mainly on how quickly receivers detect the available bandwidth and join more layers.

### 4.7.3 Packet Loss

In the Internet, packet loss may occur from transmission errors, or more commonly from network congestion. Since advances in networking technologies during the last 10 years have improved the network physical layer enormously, packet loss or corruption due to physical error is now only likely every $10^{-6}$ packets [155]. Some companies (such as *Actelis Network* [1]) even claim from their experiments that the

*Bit Error Rate (BER)* of a physical network can be as small as every $10^{-9}$, or even $10^{-15}$. Hence, the vast majority of packet loss is caused by network congestion and overflowing queues at routers or switches.

Packet loss is a waste of bandwidth and an origin of QoS degradation. A good MR-MCC protocol would act before the network becomes severely congested and drop packets. It is desirable for a good MR-MCC protocol to cause as little packet loss as possible. RLM and RLC have been revealed to cause high packet loss while both FLID-DL and PLM claim to cause less packet loss. So, our experiments are designed to compare them.

### 4.7.4 Fairness

Although an MR-MCC protocol can even provide such a high total network utilisation and almost no packet loss, it may not be a good MR-MCC protocol without fairness. This is because it may enjoy a greater share of the resources at the expense of other protocols. It can even cause starvation[5] to other competing protocols. Or, contrarily, it may be very submissive to other competing protocols, and be starved when having other protocols sharing the same network bandwidth. Hence, one of our key evaluation criteria is a fair resource distribution. So, we also consider the following senses of fairness: inter-protocol fairness, intra-protocol fairness (in particular TCP friendliness). Further details of fairness have been given in Chapter 2.

[5]By starvation, we mean there is no resource allocated to a user.

### 4.7.5 Fast Convergence

A good MR-MCC protocol should be able to allow receivers to converge rapidly from any starting state to the stable state with an optimal rate of bandwidth consumption.

### 4.7.6 Smoothness

A responsive protocol can be too sensitive to network conditions and lead to dramatic oscillation of reception rate. While this oscillation is not a problem for some applications, such as file transfer, it is a severe problem for multimedia applications that require smoothness of their reception rate. In such cases, oscillation of reception rate can badly affect the satisfaction of users. Therefore, apart from being responsive, a good MR-MCC protocol for multimedia applications must not show extreme oscillatory behaviour.

### 4.7.7 Scalability

According to [105], scalability refers to the behaviour of the protocol in relation to the number of receivers and network paths, their heterogeneity, and the ability to accommodate dynamically variable sets of receivers. The IP Multicasting model provided by RFC-1112 [47] is largely scalable, as a sender can send data to a nearly unlimited number of receivers. Therefore, good multicast congestion control mechanisms should be designed carefully to avoid severe scalability degradation. Scalability is a key property of MR-MCC (in comparison with SR-MCC), and makes it as an accepted solution for multicast congestion control for the Internet. Both FLID-DL and PLM are very scalable due to the nature of MR-MCC protocols. Furthermore, the mechanisms of both protocols completely avoid feedback from receivers back to

the sender. In addition, they do not use receivers' synchronisation messages (like RLM), which cause scalability degradation.

### 4.7.8 Feasibility

Whether the implementation of the protocol is feasible is also a key criterion. It would be better to keep the algorithms and implementation as simple as possible.

## 4.8 Experimental Design and Simulation

### 4.8.1 Simulation Tools

The ns-2 network simulation package [121] is used for our experiments. In order to simulate FLID-DL and PLM, we use ns-2 version 2.1b6a, together with FLID-DL and PLM extended modules, provided by the authors of each protocol. The introductory material on ns-2 can be found in Chapter 3. The simulation scripts used for our experiments can be found at the author's *Research Log Web Page* [136].

### 4.8.2 Performance Metrics

**Throughput**

Throughput is defined as the number of data packets (in bits) received at the receiver in a unit of time. For our experiments, the throughputs are reported in Kbps unless noted otherwise. The throughput gained by each flow indicates the rate gained and the bandwidth used by that flow. In general, congestion control is to reduce the throughput in the presence of congestion and to increase it in the absence of congestion. Also, the smoothness or oscillation of throughput with time can show the stability of rate adaptation mechanisms.

**Efficiency of Network Utilisation**

We define efficiency *(E)* of network utilisation as the ratio of throughput gained over the maximum possible throughput. *E* is actually a normalised throughput, which is bounded from zero to one. If *E* is one, then the protocol has fully utilised the available bandwidth.

**Packet Loss Ratio**

We define *Packet Loss Ratio (PLR)* as the ratio of the number of packets lost over the total number of packets transmitted during the simulation. Packet loss causes a waste of bandwidth. The higher *PLR*, the lower *E*. Furthermore, *PLR* is one of the most important sources of QoS degradation. According to Boyce and Galianello [28], the effect of *PLR* on multimedia applications (such as MPEG video sent over the Internet) can be huge. They have shown that *PLR* of 3% can cause 30% of a video stream to be unusable.

**Convergence Time**

Convergence time is the time taken by a receiver to adjust bandwidth consumption to a stable state from any starting state. Due to the non-deterministic nature of various entities, the network will not generally converge to a single static steady state. However, the network typically reaches a ‘dynamic’ stable state in which it oscillates around an optimal rate [74]. Efficient MR-MCC protocols would have a short convergence time to reach this ‘dynamic’ stable state.

**Equality Index**

The *equality index (Q)*, also known as the *fairness index*, has been introduced by Jain [81]. It is a quantitative description of fairness. $Q$ can be calculated as:

$$Q = \frac{(\sum_{i=1}^{n} T_i)^2}{n * \sum_{i=1}^{n} T_i^2} \quad (4\text{-}1)$$

where $n$ is the number of sources and $T_i$ is the throughput of the $i^{th}$ source.

According to Jain [81], $Q$ is a good metric for quantifying fairness as it holds the following properties:

1. Population size independence: the metric is applicable to any number of users.
2. Scale independence: the metric is independent of scale.
3. Bound: the metric is bounded between zero and one.
4. Continuity: the metric is continuous. Any slight change could show up in the metric.

**Intra-protocol Fairness Index**

Widmer [164] has proposed to calculate *Intra-protocol Fairness Index (I)* as:

$$I = \frac{Min(B_i)}{Max(B_i)} \quad (4\text{-}2)$$

where $B_i$ is the bandwidth consumption of flow $i$. It is the ratio of the minimum average flow throughput and the maximum average flow throughput. $I$ varies from zero to one. A zero value of $I$ indicates that at least one flow receives no bandwidth at all. A value of one is achieved in the case of an equal distribution of bandwidth.

**TCP-friendliness Ratio**

The *TCP-friendliness Ratio* (*F*) is used as a quantitative description of the TCP-friendliness. It has been used previously in several studies (such as [164], [123]). *F* can be calculated as:

$$F = \frac{T_m}{T_t} \tag{4-3}$$

where $T_m$ is the average throughput of non-TCP flows competing with TCP flows on the same link, and $T_t$ is the average throughput of TCP flows. To be able to use Eq. (4-3), $T_m$ and $T_t$ must not be zero. Otherwise, this metric cannot be applied. If *F* is one *(F =1)*, the protocol is perfectly TCP-friendly. If *F* is a lot more than one ($F >> 1$), the protocol is considered harmful to TCP. If *F* is a lot less than one ($F << 1$), the protocol is too submissive to TCP, and it would not be able to support effective transmission.

### 4.8.3 Simulation Parameters

| Parameters | Default Values |
|---|---|
| PP Burst Length | 2 |
| PP Min-required to Estimate | 3 |
| Check Period | 1 second |
| Blind Period | 0.5 second |
| Loss Threshold | 10% |
| Queuing Scheme | FQ |

**Table 4-5: PLM default parameters**

In this section, we define default parameters used in our experiments. To be compared, two sets of competing multicast protocols, namely PLM and FLID-DL are used in our simulation experiments. For the experiments that also simulate TCP connections, we use the ns-2 implementation of TCP Reno, which is the most commonly used flavour of TCP [128]. The maximum size of TCP congestion window is set to 2000 packets to remove the effect of the maximum window size. The applications on top of TCP sources are infinite FTP sessions that have unlimited data to send. The packet size of all flows (PLM, FLID-DL and TCP) is chosen to be 512 bytes.

PLM's default parameters are summarised in Table 4-5. They are set according to the recommended values in [95]. A *Constant Bit Rate (CBR)* source with a PP sending scheme is used in order to simulate each layer of a PLM source. The queuing scheme is FQ, with the size of 20 packets for each flow. To be comparable, we organise PLM layers in the same multiplicative way as used in FLID-DL.

Table 4-6 summarises important default parameters of FLID-DL used in our experiments. These parameters are set according to the recommended values from the original paper presenting FLID-DL [34]. The queuing scheme is drop-tail, with a queue size of twice delay-bandwidth product of each scenario. The minimum and maximum IGMP leave latency are set to zero. This is to be compatible with PLM modules, which do not simulate IGMP leave latency and assume the deployment of Fast IGMP [146].

| Criteria | Default Values |
|---|---|
| Rate Multiplier | 1.3 |
| Base Layer Rate | 12 Kbps (3 packets/s) |
| Time Slot | 0.5 second |
| Queuing Scheme | Drop-tail |
| Min. IGMP Leave Latency | 0 |
| Max. IGMP Leave Latency | 0 |
| Simulated RTT | Twice propagation delay |

**Table 4-6: FLID-DL default parameters**

Each simulation is run 20 times using a different *RNG* seeds. Results are averaged and quoted with respect to confidence intervals of 95%. The error bar is shown where it is appropriate. We do not show error bars when intervals are very small or negligible. Some further details of parameter settings, sensitivity test, and confidence intervals have been given in Chapter 3.

### 4.8.4 Experiment I: Response to Network Conditions

**Simulation Scenario and Objectives**

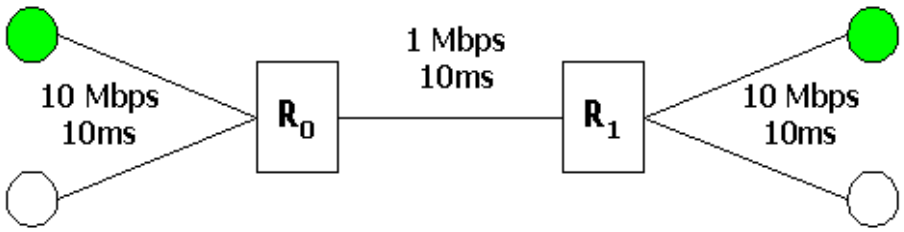



**Figure 4-1: Simulation topology of Experiment I**

The objective of this model is to compare responsiveness, fast convergence, packet loss ratio, efficiency of network utilisation and smoothness of PLM and FLID-DL when the available bandwidth changes during a session. We use a single multicast (FLID-DL or PLM) session across a bottleneck link (between router $R_0$ and router $R_1$) with 1 Mbps of bandwidth and 10 milliseconds of delay (see Figure 4-1). Each exterior link is set to 10 Mbps of bandwidth and 10 milliseconds of delay. We start the multicast source at time 0 and start its sink after 3 seconds. At time 20 seconds, we start a CBR source sharing over the bottleneck link at rate 500 Kbps to use half of the bottleneck bandwidth. At time 40 seconds, we decrease the rate of the CBR source to 250 Kbps and leave 750 Kbps available bandwidth for the multicast session. The simulation is run for 80 seconds.

**Simulation Results and Discussion**

In Figure 4-2, graphs of the throughput against the simulation time are plotted. Overall, both PLM and FLID-DL sessions can react rapidly to the change of the available bandwidth. However, from the results, PLM shows faster convergence, more responsiveness, and smoother throughput than FLID-DL.

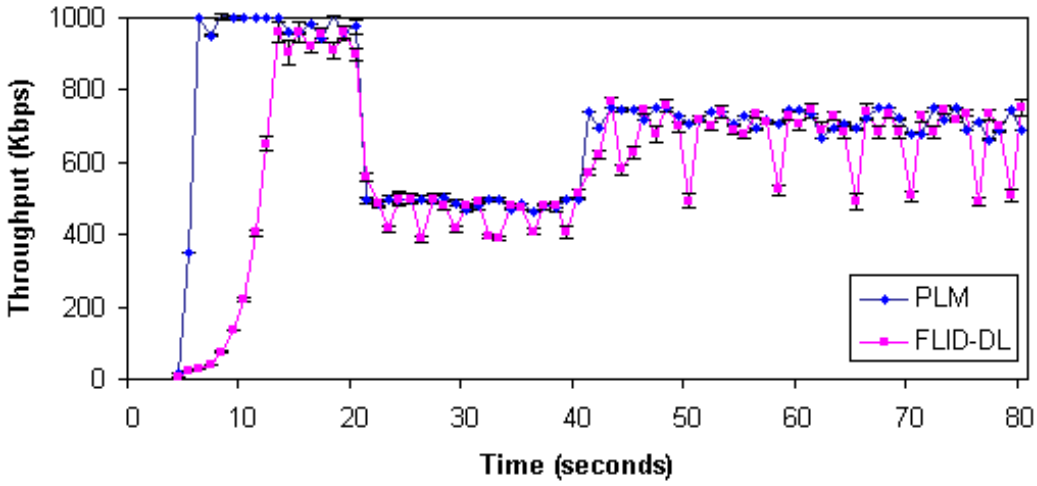



**Figure 4-2: Responding to changes in available bandwidth**

From the figure, when FLID-DL starts after 3 seconds, it takes roughly 11 seconds of convergence time to reach the optimal rate (1 Mbps). Yet, PLM can converge quicker to the optimal rate within only 3 seconds. The rate adaptation scheme of PLM also shows more responsiveness to the changes of network condition. It takes only 1 second to adjust the rate when the availability of bandwidth changes after 20 and 40 seconds, while FLID-DL takes approximately 3 seconds. This is because the rate adaptation scheme of FLID-DL allows the subscription and un-subscription only once every time slot, while PLM's rate adaptation scheme allows multiple subscriptions and un-subscriptions every check period.

Furthermore, FLID-DL is less smooth throughput compared to PLM. The oscillatory behaviour of FLID-DL is because the available bandwidths fall between two

cumulative subscription rates, as shown in Table 4-7. Hence, the reception rate of the FLID-DL receiver oscillates between the under-utilisation of the available bandwidth and over-utilisation of the available bandwidth.

| Duration | Available Bandwidth (Kbps) | Layers | Cumulative Rate (Kbps) |
|---|---|---|---|
| 21-40 | 500 | 15 | 472.49 |
| | | 16 | 614.23 |
| 41-80 | 750 | 16 | 614.23 |
| | | 17 | 798.49 |

**Table 4-7: Cumulative rate of FLID-DL vs. Available bandwidth**

By using a join experiment as its bandwidth-probing mechanism, FLID-DL can oversubscribe and cause packet loss. For example, in our experiment during 21-40 seconds, when it subscribes at 16 layers, its subscription rate (614 Kbps) exceeds the available bandwidth (500 Kbps), and it would drop to 15 layers. However, once the FLID-DL receiver subscribes at a particular subscription level, it must stay at this level for the duration of the time slot (in this case, 0.5 second) before being able to unsubscribe. During that time, the exceeded subscription rate results in packet loss.

On the other hand, by using PP as its bandwidth-probing mechanism, PLM can detect the bandwidth availability without causing any packet loss, and responds earlier to the congestion. As shown in Table 4-8, during the whole simulation, the *PLR* of PLM is 0%, while that of FLID-DL is approximately 5%.

| Metric | PLM | FLID-DL |
|---|---|---|
| Convergence Time | 3 seconds | 11 seconds |
| *PLR* | 0% | 5 % |
| Avge. Throughput | $697 \pm 4.8$ Kbps | $583 \pm 9.4$ Kbps |
| *E* | $0.94 \pm 0.01$ | $0.79 \pm 0.02$ |

**Table 4-8: Results from Experiment I**

From Table 4-8, the average throughput during the whole simulation of PLM is higher than that of FLID-DL. *E* also indicates that PLM is more efficient in utilising bandwidth compared to FLID-DL. This is because the rate adaptation scheme of PLM can make better use of the available bandwidth; also, the bandwidth-probing scheme of PLM causes less bandwidth wasted from packet loss.

### 4.8.5 Experiment II: TCP-friendliness Test

**Simulation Scenario and Objectives**

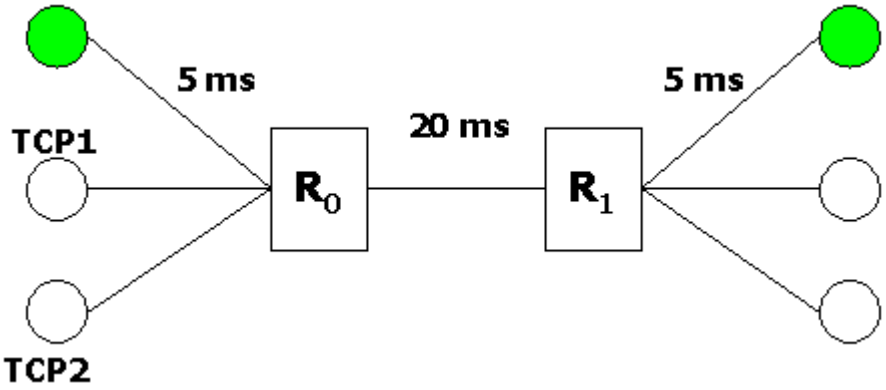



**Figure 4-3: Simulation Topology of Experiment II**

This experiment set aims at comparing inter-protocol fairness of PLM and FLID-DL, particularly the fairness towards TCP (TCP-friendliness). We use the dumbbell topology depicted in Figure 4-3 shared between two TCP connections and one multicast (FLID-DL or PLM) session. The bottleneck link between router $R_0$ and $R_1$ has a delay of 20 milliseconds. Each exterior link has a delay of 5 milliseconds. The bandwidths of bottleneck and the exterior links are varied, as shown in Table 4-9.

We run each simulation for 200 seconds. The first TCP connection (TCP1) starts at the beginning of the simulation, and at time 20 seconds, we start the multicast session (FLID-DL or PLM). At time 60 seconds, we start the second TCP connection (TCP2).

| Case | Bottleneck link's bandwidth | Each exterior link's bandwidth | Whole exterior links' demand |
|---|---|---|---|
| I | 300 Kbps | 100 Kbps | 300 Kbps |
| II | 300 Kbps | 10 Mbps | 30 Mbps |
| III | 10 Mbps | 10 Mbps | 30 Mbps |

**Table 4-9: Three cases of varying bottleneck and exterior links' bandwidth**

**Simulation Results and Discussion**

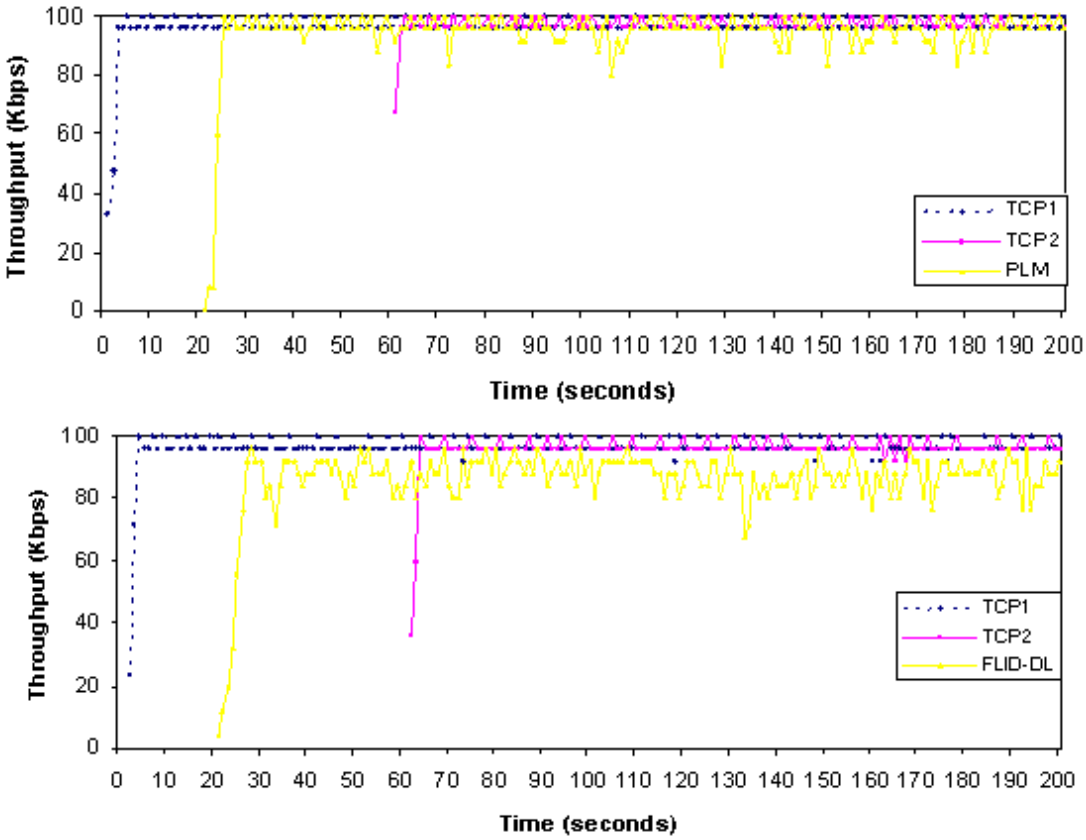



**Figure 4-4: Bandwidth share of Case I**

Figure 4-4 shows the results of *Case I* when the bottleneck link bandwidth is set to 300 Kbps and each exterior link's bandwidth is set 100 Kbps. This scenario represents the situation when the bottleneck is saturated but not yet congested. Due to no congestion, both PLM and FLID-DL obtain a fair bandwidth share with TCP flows, and show efficient bandwidth utilisation. The average throughput and the *TCP-Friendliness Ratio (F)* during the last 100 seconds of PLM and FLID-DL are shown comparatively in Table 4-10.

| Protocols | Avge. Throughput (Kbps) | | | *F* |
|---|---|---|---|---|
| | TCP1 | TCP2 | Multicast | |
| PLM | 98 ± 0.8 | 98 ± 0.8 | 96 ± 0.3 | 0.98 |
| FLID-DL | 98 ± 0.8 | 98 ± 0.8 | 87 ± 1.1 | 0.90 |

**Table 4-10: Results of Case I**

From the graphs, we can see that PLM converges faster to the optimal rate than FLID-DL. When the PLM session starts at time 20 seconds, it converges very fast to the optimal rate and shares bandwidth fairly with TCP1. The convergence time taken by PLM is only 4 seconds. On the other hand, FLID-DL shows a much slower convergence. It takes 7 seconds before FLID-DL can converge to the optimal rate.

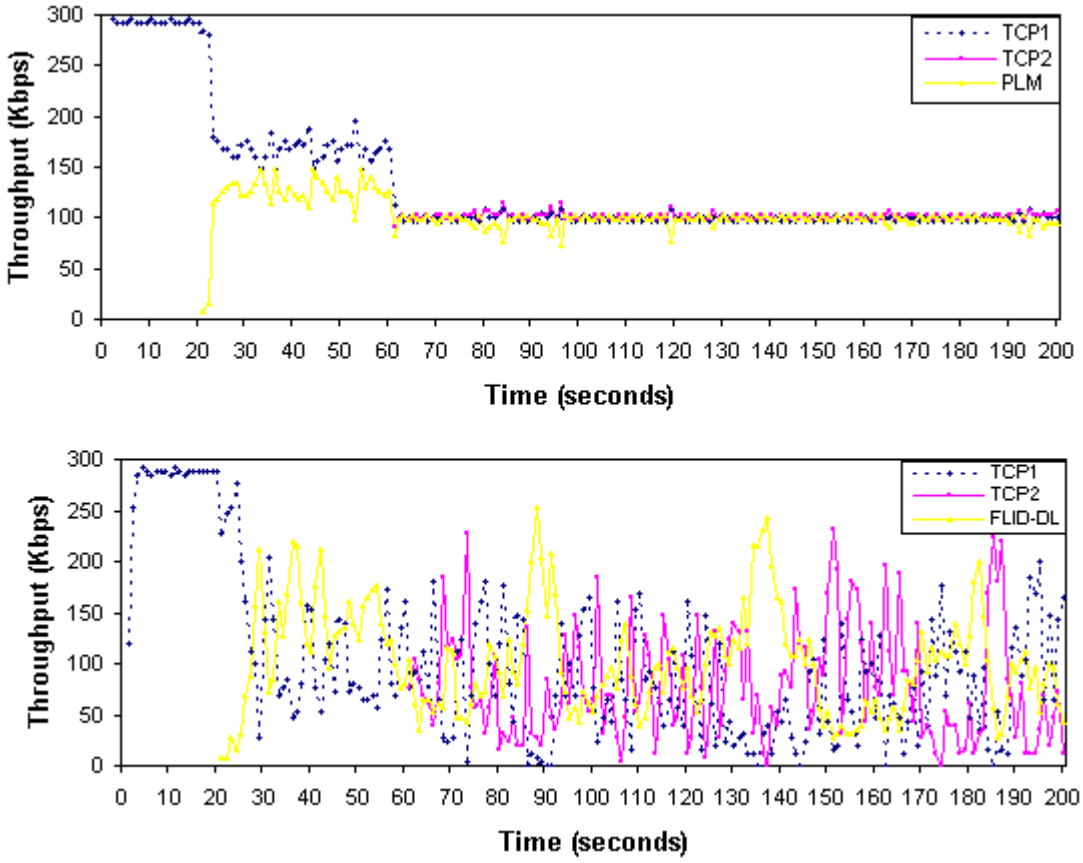



**Figure 4-5: Bandwidth share of Case II**

| Protocols | Avge. Throughput (Kbps) | | | *F* |
|---|---|---|---|---|
| | TCP1 | TCP2 | Multicast | |
| PLM | 98 ± 0.8 | 101 ± 0.8 | 99 ± 0.35 | 0.99 |
| FLID-DL | 71 ± 0.6 | 80 ± 0.7 | 95 ± 1.3 | 1.3 |

**Table 4-11: Results of Case II**

Figure 4-5 shows the results in *Case II* when we set the exterior link's bandwidth to 10 Mbps and the bottleneck bandwidth to 300 Kbps. This scenario represents the case of severe congestion.

From the figure, we can see that even under severe congestion, PLM can still behave very fairly, responsively and efficiently in sharing bandwidth with both TCP sessions. When it starts, it converges very fast to the optimal rate and shares the bandwidth with TCP fairly. FLID-DL and two TCP sessions can also adjust to be fair to each other. However, from Figure 4-5, we notice that FLID-DL shows oscillatory behaviour and causes oscillatory behaviour of TCP flows. PLM, on the other hand, shows smooth results.

Table 4-11 shows the average throughputs and *TCP-Friendliness Ratio* (*F*) during the last 100 seconds of PLM and FLID-DL comparatively. The value of *F* for PLM is 0.99 (very close to ideal TCP friendliness), while for FLID-DL it is 1.3.

Figure 4-6 shows the result in *Case III*. In this scenario, PLM still maintains its fast convergence, efficiency of bandwidth utilisation and TCP-friendliness properties very well. Conversely, FLID-DL has not shown TCP-friendliness. It is aggressive towards TCP connections. After 60 seconds, its bandwidth consumption stays at

approximately 6 Mbps, while the TCP connections can only gain less than 2 Mbps bandwidth share each.

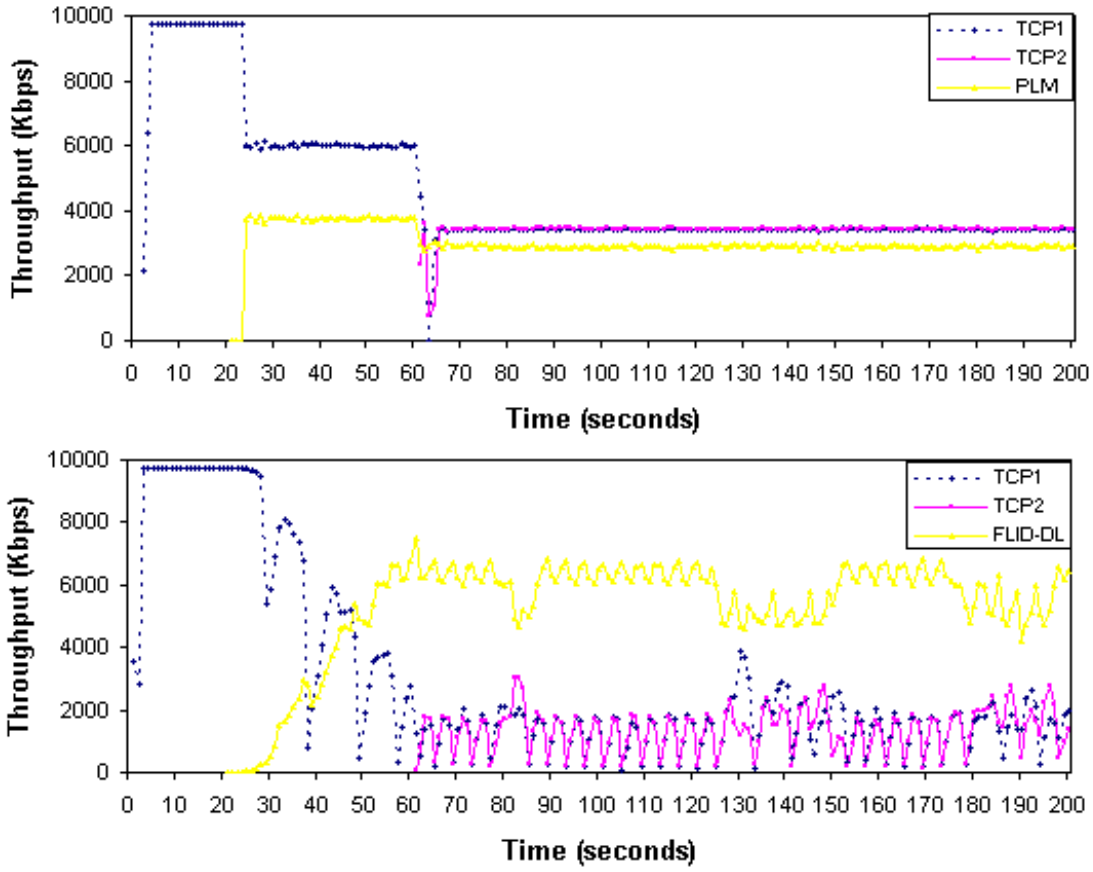



**Figure 4-6: Bandwidth share of Case III**

Table 4-12 shows the average throughputs and *Friendliness Ratio (F)* during the last 100 seconds of PLM and FLID-DL comparatively. The value of *F* for PLM is 0.8 while for *F* of FLID-DL it is 4.1 (which is harmful to TCP).

| Protocols | Avge. Throughput (Kbps) | | | *F* |
|---|---|---|---|---|
| | TCP1 | TCP2 | Multicast | |
| PLM | 3432 ± 24.4 | 3432 ± 24.4 | 2902 ± 11.7 | 0.8 |
| FLID-DL | 1483 ± 9.2 | 1367 ± 8.7 | 5837 ± 45.5 | 4.1 |

**Table 4-12: Results of Case III**

In summary, from these experiments, we have seen the TCP-unfriendliness of FLID-DL under certain network conditions. This is because FLID-DL does not take into account the RTT in its adaptation of rates. Its rate adaptation is also less responsive than TCP. Conversely, PLM can maintain fairness towards TCP by using a FQ at routers. It tries to uses bandwidth less than or equal TCP only.

### 4.8.6 Experiment III: PLM without FQ

**Simulation Scenario and Objectives**

From Experiment II, we can see that PLM is very inter-protocol friendly (in particular, TCP-friendly). This is because PLM uses FQ to enforce fairness, and assumes that FQ exists in every router. However, in terms of feasibility, FQ may not exist in routers throughout the Internet.

Hence, in this Experiment III, we reuse the same model and parameters as used in Experiment II, but run PLM without FQ. Instead of FQ, we use a drop-tail queuing scheme, which is supported by any ordinary router. The queue size for each router is set to twice delay-bandwidth product. The objective of this experiment is to study the TCP-friendliness behaviour of PLM when there is no FQ in the routers as assumed.

**Simulation Results and Discussion**

For Case I, PLM even without FQ can maintain good fairness towards TCP. The results are the same as those of PLM with FQ (shown in Figure 4-4). This is because there is no congestion here.

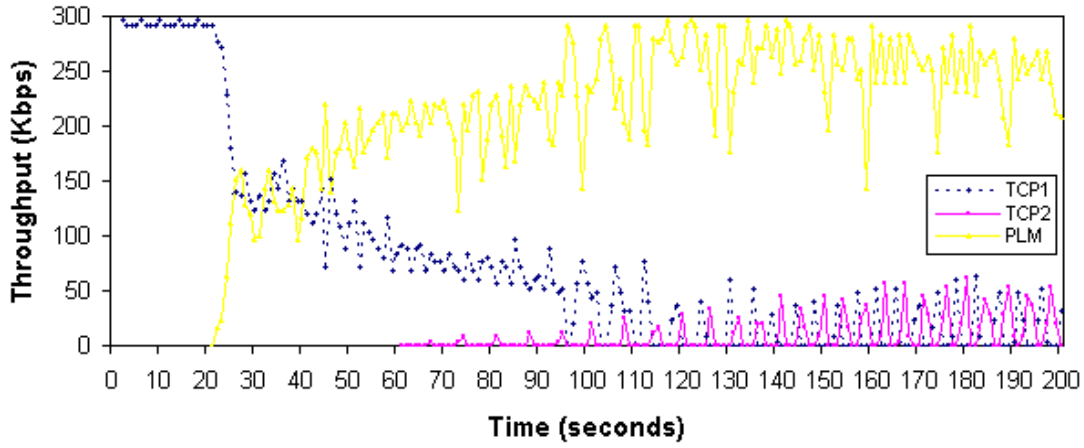



**Figure 4-7: Bandwidth share of PLM without FQ (Case II)**

Figure 4-7 shows the results of Case II when congestion is severe. From the figure, without FQ, PLM cannot maintain fairness towards TCP. Since TCP is a responsive flow, it reduces the bandwidth consumption when congestion is detected. As a result, TCP1 reduces its rate when PLM starts after 20 seconds, and again when TCP2 starts after 60 seconds. On the other hand, PLM relies on PP to detect available bandwidth and adapt its rate accordingly. When TCP reduces its rate, PLM can quickly detect that more bandwidth is available and increases its bandwidth share. So, TCP keeps decreasing its bandwidth share due to less available bandwidth from the increased rate of PLM, while PLM keeps increasing its bandwidth share until it reaches the limitation of bottleneck.

Figure 4-8 shows the results of Case III. From the plot, PLM without FQ behaves in a less TCP-friendly manner.

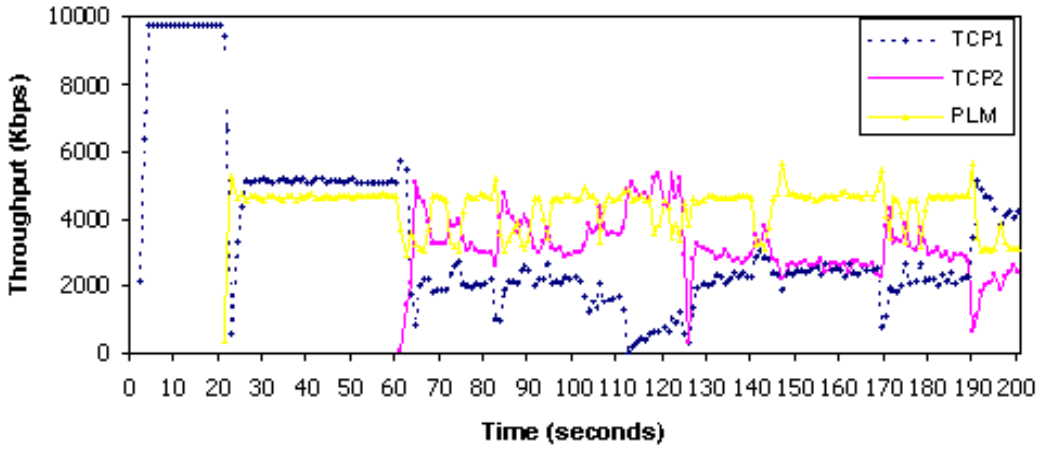



**Figure 4-8: Bandwidth share of PLM without FQ (Case III)**

| Protocols | Avg. Throughput (Kbps) | | | *F* |
|---|---|---|---|---|
| | TCP1 | TCP2 | PLM | |
| PLM without FQ in case II | 15 ± 0.83 | 12 ± 0.08 | 255 ± 0.85 | 19 |
| PLM without FQ in case III | 2177 ± 16.6 | 3105 ± 21.2 | 4357 ± 12.4 | 1.65 |

**Table 4-13: Results of PLM without FQ**

Table 4-13 shows the average throughputs and *TCP-Friendliness Ratio (F)* during the last 100 seconds of PLM with FQ in Cases II and III. The average throughputs and *F* indicate that PLM without FQ is aggressive towards TCP connections. Especially, in Case II, the *F* value of PLM without FQ is 19, indicating completely dominant behaviour towards TCP. Furthermore, the average throughputs of TCP1, TCP2, and PLM without FQ show that PLM without FQ causes starvation of the TCP sessions.

In summary, the experimental results in this section suggest that PLM cannot maintain TCP-friendliness and shows aggressive behaviour towards TCP when there is no FQ to regulate the fair share at routers as assumed.

### 4.8.7 Experiment IV: Intra-protocol Fairness Test

**Simulation Scenario and Objectives**

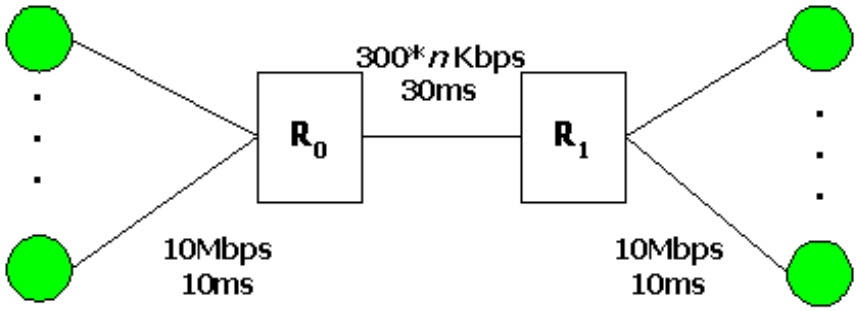



**Figure 4-9: Simulation Topology of Experiment IV**

This set of experiments aims at testing the intra-protocol fairness of PLM and FLID-DL when they compete with themselves. Ideally, each multicast session should use approximately the same amount of the available bandwidth. Figure 4-9 shows the simulation topology where multiple PLM or FLID-DL sessions consisting of $n$ sources and $n$ sinks share a bottleneck link, connected by two routers ($R_0$ and $R_1$). The number of sessions ($n$) is varied from 1 to 16. The bottleneck link between router $R_0$ and $R_1$ is set to have 30 milliseconds of delay and 300 * $n$ Kbps of bandwidth, where $n$ is the number of multicast sources. Each exterior link (between sources and router $R_0$ and between router $R_1$ and sinks) has a delay of 10 milliseconds and a bandwidth of 10 Mbps.

All multicast sessions start at a random time between time 0 and time 10 seconds. The simulation is run for 100 seconds. We calculate the average throughput, *Equality Index* (*E)* and *Intra-protocol Fairness Index (I)* of each session for the last 50 seconds.

**Simulation Results and Discussion**

| *n* | *E* | | *PLR (%)* | | *Q* | | *I* | |
|---|---|---|---|---|---|---|---|---|
| | **PLM** | **FLID** | **PLM** | **FLID** | **PLM** | **FLID** | **PLM** | **FLID** |
| 2 | 0.92 | 0.92 | 0 | 16 | 0.99 | 0.99 | 1.00 | 0.99 |
| 3 | 0.88 | 0.98 | 0 | 12 | 0.99 | 0.99 | 0.99 | 0.94 |
| 4 | 0.88 | 0.98 | 0 | 11 | 0.96 | 0.99 | 0.99 | 0.90 |
| 5 | 0.88 | 0.98 | 0 | 11 | 0.95 | 0.99 | 0.99 | 0.91 |
| 6 | 0.89 | 0.98 | 0 | 11 | 0.97 | 0.96 | 0.99 | 0.94 |
| 7 | 0.90 | 0.98 | 0 | 11 | 0.97 | 0.96 | 0.99 | 0.92 |
| 8 | 0.91 | 0.98 | 0 | 12 | 0.98 | 0.99 | 1.00 | 0.91 |
| 9 | 0.91 | 0.98 | 0 | 12 | 0.99 | 0.99 | 1.00 | 0.88 |
| 10 | 0.91 | 0.98 | 0 | 12 | 0.98 | 0.99 | 1.00 | 0.94 |
| 11 | 0.92 | 0.98 | 0 | 11 | 0.96 | 0.99 | 0.99 | 0.86 |
| 12 | 0.91 | 0.98 | 0 | 11 | 0.98 | 0.99 | 1.00 | 0.83 |
| 13 | 0.92 | 0.98 | 0 | 11 | 0.97 | 0.95 | 0.99 | 0.86 |
| 14 | 0.92 | 0.98 | 0 | 11 | 0.97 | 0.95 | 1.00 | 0.86 |
| 15 | 0.92 | 0.98 | 0 | 10 | 0.97 | 0.99 | 0.99 | 0.87 |
| 16 | 0.93 | 0.98 | 0 | 10 | 0.98 | 0.99 | 1.00 | 0.84 |

**Table 4-14: Intra-protocol fairness of PLM and FLID-DL**

The *Equality Index (Q)* and *Intra-protocol Friendliness Index (I)*, in Table 4-14, show that both PLM and FLID-DL are intra-protocol friendly. The *Q* values of both protocols are similar. However, as indicated by *I,* PLM shows better intra-protocol friendliness.

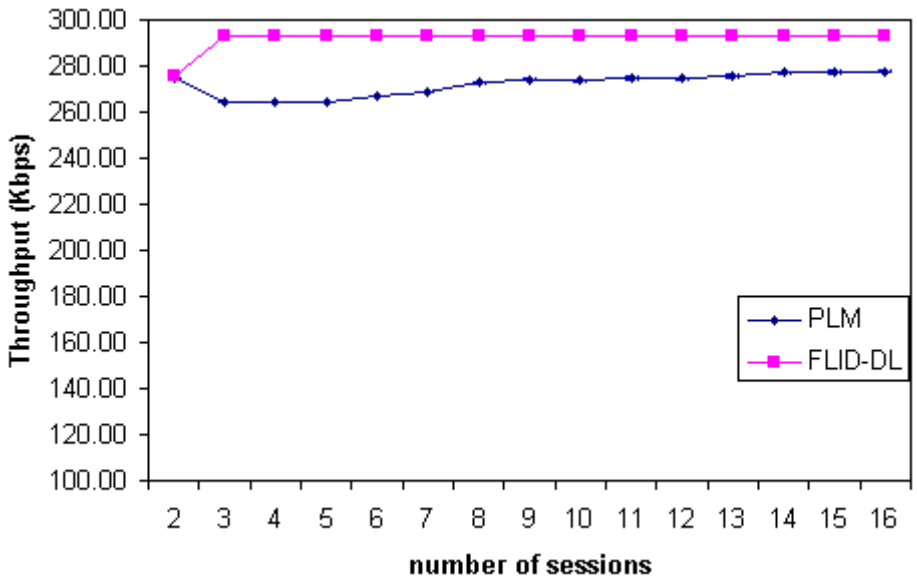



**Figure 4-10: Average throughput during last 50 seconds**

Figure 4-10 shows the average throughputs during the last 50 seconds. From the plots, FLID-DL gains higher average throughputs. Also, as shown in Table 4-14, FLID-DL has higher *E* values. This is because the layer granularity and the bottleneck available bandwidth do not match; also PLM's rate adaptation algorithms are conservative. In this experimental scenario, the available bandwidth is 300 Kbps, while the subscribed rates of 13 layers and 14 layers are 279.58 and 363.45 Kbps, respectively. PLM subscribes to a layer only when the subscribed bandwidth is less than the estimated bandwidth.

So, the PLM sessions have never subscribed more than 13 layers, and under-utilise the bandwidth. On the other hand, FLID-DL's subscribed level oscillates between the under-utilised (13 layers) and the over-utilised (14 layers) levels. This makes FLID-DL gain higher average throughput than PLM in this situation. However, this comes with the expense of the higher PLR and the oscillation of reception rates. During the whole simulation, the PLRs of FLID-DL vary from 10% to 16%, while PLRs of PLM are always 0%.

## 4.9 Discussion

From the experiments, our simulation results reveal that PLM performance is better in terms of smoothness, responsiveness and fast convergence, TCP-friendliness, and low packet loss rate. This is because:

- The PP is a better congestion detection engine compared with the join experiment that FLID-DL inherited from RLC and RLM. PP hardly causes packet loss. It can detect congestion before the network becomes severely congested.
- PLM's rate adaptation mechanisms are quicker than the join experiment at every SP of FLID-DL. It also provides smoother throughput.
- The FQ used in PLM can efficiently ensure fairness, especially TCP-friendliness. FLID-DL mechanisms in contrast are not enough to ensure TCP-friendliness. Hence, FLID-DL exhibits unfair behaviour towards TCP under certain network conditions.

However, the feasibility of PLM is still arguable, as it assumes a fair scheduler at the routers. Without a fair scheduler, our experimental results have shown that PLM can become very aggressive and cannot maintain its TCP-friendliness any more.

For the problem of IGMP leave latency, PLM has no proposed solution, but relies on Fast IGMP. Without the deployment of Fast IGMP, IGMP leave latency can cause less responsiveness, higher *PLR*, and less network utilisation to PLM. For FLID-DL, IGMP leave latency can be mitigated by its DL mechanisms. So, IGMP leave latency would have less impact on FLID-DL, even no deployment of Fast IGMP. Nevertheless, while mitigating the IGMP leave latency, DL poses the problem of an

excessive number of control messages. In addition, an argument is whether this problem would be solved at congestion control protocol or at the IGMP protocol.

From this work, we have learnt that using a simple mechanism (like PP) for explicit rate adjustment could be a better solution to deal with the congestion control problem than relying on packet loss to detect congestion (like join experiment). The available bandwidth estimation is an explicit way of congestion notification. It gives a chance to avoid causing congestion or fixing it at the incipient state while the packet loss would only detect congestion after it has occurred. PP would provide a better solution to the congestion control problem than *join experiment* because *congestion avoidance* is better than *congestion recovery*.

## 4.10 Summary

We have reviewed the mechanisms of PLM and FLID-DL. PLM uses a PP technique to estimate the available bandwidth and relies on a fair scheduler to enforce fairness, whereas FLID-DL uses a probabilistic synchronised join experiment to detect congestion, and relies on the distribution of layers to enforce TCP-friendliness. Because both FLID-DL and PLM claim a substantial improvement over RLM and RLC in terms of network utilisation, responsiveness, low packet loss rate, smoothness, TCP-friendliness, and fast convergence, we have conducted a simulation-based performance evaluation to compare them.

Our simulation results have shown a few advantages of PLM over FLID-DL especially the way of PLM tackles the congestion control problem through explicit rate adjustment. However, PLM uses the FQ, which may be unfeasible to enforce inter-protocol fairness.

# Chapter 5

# ERA: Rationale and Design

## 5.1 Introduction

In recent years, several studies (such as [34], [88], [95], [102], [113], [151], [159] and [169]) have focused on the design of MR-MCC protocols. In the previous chapter, we have investigated some recently proposed MR-MCC protocols and evaluated their advantages and disadvantages. We have found that the previous proposals have some major drawbacks. Some designs cause over-subscription and high packet losses. Some are slow to converge and unresponsive. Some are TCP-unfriendly. Some designs are too complex or even arguable in terms of feasibility. Others are not scalable.

Hence, in this chapter, we propose a new design of MR-MCC, which has the following properties: scalability, responsiveness, fast convergence, fairness (including intra-session fairness, intra-protocol fairness, inter-protocol fairness and TCP friendliness), efficiency in network utilisation, and simplicity to implement. Our design is based on an estimation of an explicit target rate using a *Packet-pair Probe (PP)* and a TCP throughput equation. By combining this target rate estimation, the receiver-driven layered multicast approach and our new rate adjustment algorithms as well as our framework for the cooperation between the sender and the receivers, we contribute an innovative MR-MCC protocol, called *Explicit Rate Adjustment (ERA)*.

The remainder of this chapter is organised as follows. Section 5.2 gives the rationale for this work. In Section 5.3, we describe our design goals. Section 5.4 explains the protocol basics. The framework and algorithms of ERA are proposed in Section 5.5. In Section 5.6, we illustrate protocol implementation. The discussion of design arguments (such as IGMP leave latency, security issues, and packet-pair probe) is included in Section 5.7. Finally, in Section 5.8, we summarise.

## 5.2 Rationale for Research

Since 1996, a few studies have been conducted and aimed at providing good MR-MCC protocols. However, in Chapter 4, we have examined some of the previous MR-MCC proposals (RLM, RLC, FLID-DL and PLM), and found out several deficiencies that need to be improved. Some major deficiencies can be described as follows:

- **High PLR**: For some MR-MCC proposals (such as RLM, RLC, and FLID-DL), the congestion detection relies on the detection of packet loss (such as by using the variant versions of join experiment). So, they fundamentally cause high PLR, and as mentioned in Chapter 4, high PLR degrades quality of service.
- **Slow convergence and inefficiency in utilising network**: From our performance studies of MR-MCC previous proposals in Chapter 4, we have found that RLM, RLC and FLID-DL slowly converge to the optimal layer; thus leave network bandwidth under-utilised.
- **Slow responsiveness:** We have also found that rate adaptation mechanisms of RLM, RLC and FLID-DL are too slow, and cause the protocols to respond to the network congestion slowly. This slow response results in congestion persistency and high packet loss rate.
- **TCP-unfriendliness:** Our investigation in Chapter 4 has revealed the TCP-unfriendliness of RLM, RLC, and FLID-DL. In competing with TCP connections, it can cause a starvation of those TCP connections.
- **Unfeasibility of implementation scheme**: Unlike RLM, RLC and FLID-DL, PLM have not suffered from the above deficiencies (High PLR, slow convergence, slow responsiveness, TCP-unfriendliness). However, the implementation scheme of PLM is unfeasible due to the use of FQ at every

router to enforce the fairness. Without FQ, our experiments in Chapter 4 have revealed that PLM cannot maintain its fairness (including TCP-friendliness) property.

As a result, in this chapter, we propose an innovative design of MR-MCC to conquer these deficiencies.

## 5.3 Design Goals

Our goal is to create a layered multicast congestion control protocol that has the following properties:

### 5.3.1 Responsiveness

Basically, an available bandwidth on a network changes constantly. It increases or decreases as end nodes and links go up or down. Our MR-MCC should dynamically match the bandwidth demand to the available bandwidth. Hence, it should allow users to increase the demand when additional bandwidth is available, and decrease it when the available bandwidth drops. In particular, the responsiveness to detect and fix congestion is generally the first goal of congestion control protocols. Our design goal is to detect congestion at the incipient state and quickly react by unsubscribing from layers.

### 5.3.2 High Network Utilisation

Being responsive would also provide high network utilisation. When the network bandwidth becomes available, our protocol should not leave it under-utilised.

### 5.3.3 Fast Convergence

As mentioned in Chapter 4, fast convergence is highly important for MR-MCC protocols to gain high network utilisation. Hence, our protocol is designed to allow receivers to converge rapidly from any starting state to the stable state with an optimal rate of bandwidth consumption.

### 5.3.4 Scalability

Our design goal is to provide a multicast congestion control protocol for the Internet environment. So, we want a protocol that is scalable to a nearly unlimited number of receivers, network paths and receivers' heterogeneity. Our protocol is therefore designed carefully to avoid techniques that may cause severe scalability degradation. In particular, we avoid any messages from receivers back to the sender or any messages among receivers that cause a scalability problem in RLM and other MR-MCC protocols. The goal of scalability is that the sender is insensitive to the number of receivers and multicast sessions.

### 5.3.5 Fairness

Fairness may not be a big problem during low traffic load when demand of all competing connections can be satisfied. However, when the network becomes congested (i.e., the available bandwidth is less than the demand), it is crucial that the available bandwidth is allocated fairly among competing connections. Hence, fairness is one of the most significant goals of designing congestion control protocols. In our design, we consider the following senses of fairness: inter-protocol fairness (particularly TCP-friendliness), intra-protocol fairness and intra-session fairness. The details of these senses of fairness are described in Chapter 2.

### 5.3.6 Low Packet Loss Rate

As mentioned in Chapter 4, packet loss is a waste of bandwidth and can lead to the quality of service degradation. So, having low packet loss rate is one of our goals.

### 5.3.7 Feasibility and Simplicity of Implementation

The mechanisms used in our design must be feasible to implement. We also try to keep our MR-MCC algorithms as simple as possible. Our simplicity requirement is that our new multicast control scheme would be easy to specify and implement.

### 5.3.8 Supporting Various Application Natures

Some MR-MCC protocols are proposed to some specific applications only. Yet, the ERA proposal aims at providing a congestion control mechanism that can be adapted for various kinds of applications. Different applications usually have different requirements for congestion control. For instance, congestion control for reliable content delivery applications may want to use all available bandwidth, and radically reduce reception rate when there is competing traffic. In contrast, congestion control for streaming live multimedia applications may prefer to maintain a constant rate rather than try to use all available bandwidth. This is to ensure the smoothness of user reception. Also, it may not reduce reception rate as quickly as congestion control for reliable content delivery when there is competing traffic.

## 5.4 Protocol Basics

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

### 5.4.1 Best-effort Service

ERA is designed only for the *Best-Effort (BE) Service* networks, which have no quality of service guarantee. So, the multimedia applications supported by ERA will be limited to BE service only. We note that some multimedia applications may need QoS support. They therefore may need MR-MCC designs based on the assumption that QoS will be deployed in the future Internet. *Receiver-driven Layered Multicast with Priorities (RLMP)* [66], *Network-driven Layered Multicast (NLM)* [82], and *Differentiated Services Layered Multicast (DSLM)* [153] are examples of such MR-MCC designs.

### 5.4.2 Multicast Support at the Network Layer

ERA assumes multicast support at the network layer. One of two current models of multicast delivery at network layer (the Any-Source Multicast (ASM) defined in RFC-1122 [47] or the Source-Specific Multicast (SSM) defined in RFC-3569 [27]) may be used.

### 5.4.3 Single Data Source

ERA is a one-to-many congestion control protocol. We assume that all data are sent from a single source. So, ERA's congestion control is done per source. Yet, multiple data sources can be supported by running multiple instances of ERA.

### 5.4.4 Layered coding and Receiver-driven Approaches

ERA is designed by using the receiver-driven layered multicast approach (like other MR-MCC) to provide scalability for a very large heterogeneous group of receivers. We choose design options carefully to be compatible with the *Layer Coding Transport (LCT)* defined in RFC-3451 [105], which is a standard of *Layering Congestion Control (LCC)* approach.

### 5.4.5 Error Control

To support reliable multicast application, we expect an error control used together with our congestion control. A complete protocol instantiation may include a scalable error control that is compatible with the layered encoding concept [105]. Such possible error control would be the FEC approach. Its standard is defined in [104] and [106]. An effective FEC algorithm (such as DF [32] or tornado [103]) may be used together with our MR-MCC protocol. However, it is beyond the scope of this research to design effective FEC algorithms or any other multicast error control algorithms. An overview of FEC and other alternatives to multicast error control have been described in Chapter 2.

### 5.4.6 Explicit Rate Adjustment

We believe that finding a simple mechanism for explicit rate adjustment would simplify the congestion control problem. According to our algorithms, the receiver adjusts its reception rate to the target rate, which is explicitly calculated as the minimum of the estimated available bandwidth and the estimated TCP-friendly rate. The reason behind this explicit rate is that: (1) to avoid causing network congestion, we should not abuse the bandwidth by using more than an available bandwidth; (2) to

be TCP-friendly, we also should not utilise more bandwidth than TCP traffic in the same condition. The details of estimating the available bandwidth and the TCP friendly rate are described in the next section.

## 5.5 Framework and Algorithms

### 5.5.1 Sender Operation

The sender has the responsibility to encode the data into multiple layers. Then, the encoded data packets of each layer are sent as a pair to the receivers. This packet-pair will be used in order to estimate the available bandwidth at the receiver side.

The header format of each packet is shown in Table 5-1. Each field can be described as follows. *Object Identifier (OID)* identifies which object the packet contains data for. *Layered Identifier (LID)* identifies which layer the packet is a part of. *Packet Sequence Number (PSN)* is used in order to detect packet losses. *Sender Current Time (SCT)* indicates the time when the packet is sent from the sender. *First Packet-pair Flag (FPF)* indicates the first packet of the packet-pair.

| Name | Description |
|---|---|
| OID | Object IDentifier |
| LID | Layer IDentifier |
| PSN | Packet Sequence Number |
| SCT | Sender Current Time |
| FPF | First PP Flag |

**Table 5-1: Packet header format**

For every predefined *Announcing Time* ($t_{announce}$), the sender advertises a *Session Announcement Message (SAM)* to the receivers. SAM provides a session description with the following information: data rate of each layer, number of layers, IP address

of the sender, IP address and port number of each layer, packet size, object length and *Rate Adaptation Interval (RAI),* which is the predefined time interval for the receivers to adapt their reception rate. This RAI can be tuned according to the nature of application running on top ERA. Some applications (such as reliable content delivery) may prefer a short RAI to gain more responsiveness, while others (such as multimedia application) may prefer a long RAI to gain more smoothness of reception rate.

### 5.5.2 Receiver Operation

The receiver has to receive a *SAM* and interpret the session description before joining a session. After that, the receiver has a role to decode, and obtain the necessary data packets to reproduce the object. Congestion control is done at the receiver side using the algorithms in the next section.

### 5.5.3 Rate Adaptation Algorithms

*Rate Adaptation Algorithms* of ERA can be summarised as follows:

1. For every arrival of a packet-bunch, the receiver estimates the available bandwidth ($R'_{pp}$) using the technique mentioned in Section 5.5.5. If the subscribed rate is higher than $R'_{pp}$, the receiver will immediately reduce its reception rate to avoid overloading the network.
2. For every *RAI*, the receiver calculates an estimated bandwidth $R_{pp}$ as the minimum $R'_{pp}$ during the last *RAI.* There may be a pathological case, when packet bunches are lost during severe congestion. Then, we may not have enough $R'_{pp}$ to make a good estimation of available bandwidth ($R_{pp}$). In this case, we set $R_{pp}$ to –1 to indicate severe congestion.

3. The receiver also calculates *PLR, RTT, and a TCP-friendly Rate* using the techniques mentioned in Sections 5.5.6 - 5.5.8. Let *PLR = l, RTT =* $t_{RTT}$ *, and* the TCP-friendly rate = $R_{TCP}$
4. The receiver calculates its current subscribed rate ($R_i$) using Eq. (5-1) with respect to the number of subscribed layers ($i$) maintained at the receiver, and a data rate of each layer obtained from the session description.
5. The receiver estimates the target reception rate ($R_{TARGET}$) as follows:

```
If  (l > 0) Then
   If (Rpp ≥ 0) Then
         Set  RTARGET = Min(RTCP , RPP )
   Else If (Rpp = -1) Then
         Set  RTARGET = RTCP
   End If
Else
   Set  RTARGET = RPP
End if
```

6. The receiver subscribes to or unsubscribes from layers according to the $R_{TARGET}$ as follows:

```
If (Ri > RTARGET) Then
   Repeat Until (Ri ≤ RTARGET)
        If i > 0 Then
           Unsubscribe from a layer
           i = i -1
        Else
           EXIT the session
        End If
   Loop
Else If (Ri < RTARGET)
   Do While (Ri+1 < RTARGET)
        Subscribe to an additional layer
        i = i +1
   Loop
Else If (Ri = RTARGET)
   Maintain the current subscription level
End If
```

### 5.5.4 Layering

Layered encoding was proposed in [113]. It is based on the ability of a sender to generate the same data at different rates over multiple multicast streams. The sender organises multiple multicast groups into logical layers. There are still several open questions of MR-MCC design about layering as follows:

(1) **Cumulative or non-cumulative organisation of layers:** Cumulative layering means each layer provides refined information to the previous layers, and the receiver must subscribe to all layers up to and including the highest layer. For non-cumulative, each layer is independent. Receivers can choose to subscribe to any layer or only one layer. The non-cumulative scheme is also called *Simulcast* as the source transmits multiple copies of the same data simultaneously at different rates. In general, cumulative layering is used due to the complexity of framing application-level data to be compatible with non-cumulative layers and performance penalty of providing non-cumulative layering. However, the recent development of fast FEC encoding for reliable multicast (such as, [101] and [103]) for reliable multicast and fine-grained rate video coding have mitigated the problems. In addition, Byers et al. [31] suggests that a careful design of non-cumulative layering and corresponding congestion control mechanisms would allow receivers to perform *fine-grained* congestion control (that cumulative layering cannot do). However, there is only little initial work on non-cumulative layering. Whether cumulative or non-cumulative layering would be a better choice for MR-MCC is still an open question.

(2) **Layer granularity**: the layer granularity refers to the rate of each layer. Some open questions related to layer granularity are: "how many layers would be used?", "how big would each layer be?"; "fine-grained or coarse-grained?" This is actually an argument of a trade-off between the number of layers, the extra complexity introduced and the bandwidth utilisation achieved. A small number of layers would lead to a coarse-grained rate adaptation, while a large number of layers would lead to extra complexity in multicast group management, but fine-grained rate adaptation. Nevertheless, for multimedia applications, layer granularity may not have much choice because it depends highly on the CODEC used to encode audio/video. In particular, the perceived quality and the requirement of bandwidth are the key to layer organisation. Too fine-grained adjustment may be useless if that fine granularity cannot improve the user's satisfaction.

(3) **Layering scheme:** The layering scheme can be specified as *equal, double,* or *multiplicative* [34]. Some MR-MCC proposes to use doubling scheme (such as RLM [113], RLC [159]; some use multiplicative scheme (such as FLID-DL [34]). It is still an open research issue of MR-MCC what the best layering scheme would be.

For the design of ERA, we choose the layer organisation to be cumulative due to the complexity and performance cost of non-cumulative layering. All receivers must subscribe to or unsubscribe from layers in a consecutive order. If $L_j$ denotes the data rate of layer *j,* the cumulative rate ($R_i$) of a receiver, which subscribes to layer *i,* can be calculated as:

$$R_i = \sum_{j=0}^{i} L_j \quad (5\text{-}1)$$

For the layer granularity and layering scheme, we argue that none of the layering schemes is the best in every situation. The layering scheme and layer granularity should be chosen according to the application's requirement. We therefore leave the layer granularity and layering scheme of ERA unspecific. In the real-life implementation, ERA can be implemented using any suitable layering scheme and layer granularity up to its applications. The analysis of layer granularity and layering scheme towards different kinds of applications is beyond the scope of this research, and is left for future work.

### 5.5.5 Available Bandwidth Estimation

To estimate the available bandwidth, we use the *receiver-side Packet-pair bunches Probe* of Paxson [129], which is improved from the original *Packet-pair Probe* of Keshav [86].

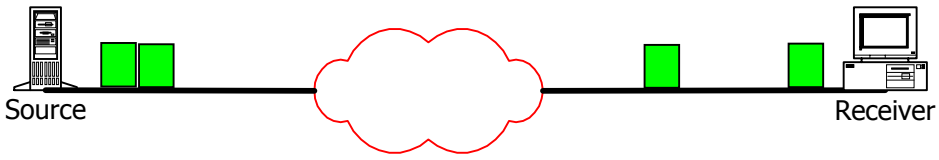



**Figure 5-1: Packet-pair Probe**

With the PP technique (illustrated in Figure 5-1), the sender in our protocol periodically sends a pair of its data packets as a burst to infer the bandwidth share of the flow. For each arrived packet, the receiver checks *FPF* to determine the first packet of the packet-pairs. According to Keshav [86], the receiver can estimate the available bandwidth ($R'_{pp}$) as:

$$R'_{PP} = \frac{8M}{t_{gap}} \qquad (5\text{-}2)$$

where $M$ is the packet size (in bytes), and $t_{gap}$ is the inter-arrival time (in seconds) of packets.

### 5.5.6 Packet Loss Rate Estimation

The PLR is calculated from the number of packets lost at the receiver divided by the number of packets sent by the sender during a certain observation period. For our protocol, the number of lost packets can be detected by checking the gap in the *PSN* field of the packet header. The number of packets sent can be estimated as the difference between the highest and lowest *PSN* during the observation period.

### 5.5.7 Round Trip Time Estimation

RTT is required for the *TCP throughput equation* (see also Eq. (5-6)). There are several alternatives proposed to estimate RTT multicast. Some possible alternatives are described as follows:

**Use RTT-request packet**

A receiver sends an RTT-request packet to the sender. Then, the sender replies immediately with an RTT-reply packet. Finally, the RTT can be estimated as the time difference between sending the request and receiving the reply. This alternative works well for unicast but faces a scalability problem in multicast. In case of multicast with a large number of receivers, the RTT-request packets sent by the receivers can overload the sender and cause *network implosion*. So, some kind of suppression technique must be used to apply this technique to multicast.

**Estimate RTT as twice one-way latency**

The sender transmits a control message every predefined period with a timestamp (i.e., an *SCT* field in our packet header) to the receivers. When the control message arrives, the receiver estimates half of RTT, as the time difference between *SCT* and the message arrival time. However, one-way latency is not a good estimation of half RTT as revealed in [43] and [130]. In particular, this method does not work for asymmetrical paths, and requires some kind of synchronisation of the clocks between receivers and senders.

**Estimate RTT in layered multicast**

Luby et al. [102] have recently proposed to estimate RTT as the difference between the time of issuance of join request and the arrival time of the first packet of the layer. However, this is not an accurate RTT either. This is because the join-request messages only propagate back to the router closest to the sender only (not the sender). So, the latency between the sender and the closest router has not been counted.

We leave the efficient RTT estimation for future work, and assume that the receiver has an efficient estimated RTT. For the real life implementation of ERA, we would suggest the technique of Luby (which is the best at the moment) until there are any better estimation techniques.

For the purpose of implementation in ns-2, we simply calculate RTT as:

$$t_{RTT} = (2 * \textit{one}\ \text{way latency}) + \varepsilon \qquad (5\text{-}3)$$

where $\varepsilon$ is an estimation of queuing delay. $\varepsilon$ is arbitrary specified just for simulation purpose. Also, we use only symmetrical paths in our simulation, and assume synchronised clock between receivers and senders.

### 5.5.8 TCP-friendly Rate Estimation

There have been several analytical and empirical studies to estimate the throughput of TCP in steady state. The first model for TCP throughput has been presented in [107]. From this model, the steady throughput (in bps) of a TCP connection ($R_{TCP}$) is given as:

$$R_{TCP} = \frac{8cM}{t_{RTT}\sqrt{l}} \tag{5-4}$$

where $c$ is a constant (varying from 0.87 to 1.31, depending on the assumption of periodic or random loss event [107]), $M$ is the packet size (in bytes), $t_{RTT}$ is the RTT (in seconds), and $l$ is the PLR (between 0.0 and 1.0).

The model makes an assumption that TCP experiences windows reduction events only because of triple duplicate ACKs, not because of timeouts. As revealed in [124], this assumption is reasonable only for low loss rate (below 0.16). However, in a higher loss rate situation, the TCP congestion control becomes more dominated by timeout events. Consequently, the model can overestimate the TCP throughput.

Hence, Padhye et al. [124] have proposed a better model for a broader range of network conditions. The model is given as:

$$R_{TCP} = \frac{8M}{t_{RTT}\sqrt{\frac{2bl}{3}} + t_{RTO}\,Min\,(1,3\sqrt{\frac{3bl}{8}})\,l(1+32l^2)} \tag{5-5}$$

where $b$ is the number of packets acknowledged by each ACK, $t_{RTO}$ is the TCP retransmission timeout (in seconds). This model is for TCP Reno. It is the TCP throughput model most widely accepted by the Internet research community.

To calculate the TCP-friendly rate in our algorithms, we simplify Eq. (5-5) as recommended in [166] by assuming: $t_{RTO} = 4 * t_{RTT}$, $b = 1$, and $Min\ (1,\ 3\sqrt{\frac{3l}{8}}) = 3\sqrt{\frac{3l}{8}}$.

Then, we get:

$$R_{TCP} = \frac{8M}{T_{RTT}\sqrt{\frac{2l}{3}} + 12\sqrt{\frac{3l}{8}}l(1 + 32l^2)} \qquad (5\text{-}6)$$

So, our TCP-friendly rate is relying on the Reno flavour, which is the most commonly used flavour of TCP [128]. Actually, TCP SACK, proposed in [111], has an improved loss recovery scheme and its deployment is now increasing [20]. However, to the best of our knowledge, there is no well-known and widely accepted model for it. Nonetheless, any improved TCP throughput model in the future can replace Eq. (5-6) in our implementation.

### 5.5.9 Receiver Coordination

The coordination of the receivers under the same bottleneck link is necessary to obtain intra-session fairness. In particular, as revealed in [113], this co-ordination is significant for MR-MCC to utilise bandwidth efficiently and handle congestion properly.

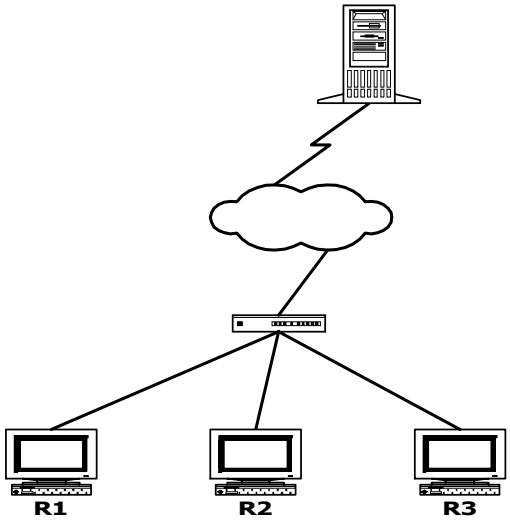



**Figure 5-2: Receivers Coordination**

As illustrated in Figure 5-2, if a receiver (for example, *R1*) unsubscribes from a layer to tackle congestion, the other receivers under the same bottleneck link (*R2* and *R3* in this case) should also unsubscribe from that layer. Otherwise, the multicast tree of that layer will not be pruned. The bandwidth consumption for that layer therefore does not cut out. This finally results in congestion persistency.

Furthermore, the subscription coordination is also important to an efficiency of bandwidth utilisation. If a receiver (for example, *R1* in Figure 5-2) subscribes for a layer, a multicast tree is grafted for this layer and consumes a certain amount of bandwidth. The other receivers under the same bottleneck link (*R2* and *R3* in this

case) should also subscribe to that layer to utilise this tree as well. Otherwise, the bandwidth used for the multicast tree is not efficiently utilised.

ERA provides coordination by relying on *Session Announcement Message (SAM)* and *Rate Adaptation Intervals (RAI)* as follows. The source sends a *SAM* at every predefined time interval ($t_{announce}$) to provide information about the transmission session (details in Section 5.5.1). *RAI* is a part of the information provided in *SAM*. In our design, we enforce that $t_{announce} = n * RAI$, where n is a positive integer. Furthermore, the receiver can join a transmission session only after it receives a *SAM*, and will adapt its reception rate every *RAI*. This helps coordinate the subscriptions and unsubscriptions among receivers since they adapt their rate at the same time.

## 5.6 Protocol Implementation

We have implemented ERA on ns-2 version 2.1b6a, which was the most updated version by the time that we started our implementations (November 1999). It is also the most widely deployed ns-2 version for multicast congestion control research as evidenced by [34], [94], [95], [147], [166] and [169]. The ns-2 modules of ERA are implemented mainly in oTcl and partly in C++. ERA has not been built from scratch but we build it on top of several previous studies. Our implementation of MR-MCC frameworks is done from our experience learning from several previous implementations of MR-MCC protocols in ns-2, such as RLM [113], RLC [159], FLID-DL [34] and PLM [95]. We should be particularly grateful to McCane for providing the first model of MR-MCC protocols and RLM's module in ns-2, and Vicisano, Luby and Legout for making RLC, FLID-DL and PLM ns-2 modules available publicly. All these previous modules are very helpful as a guideline for our own ns-2 implementation of ERA algorithms.

The complete source codes (both C++ and oTcl) and a manual of ERA are provided publicly online at [136] in order to be validated and extended further by the research community. We also provide a validation of ERA in Appendices B and C.

## 5.7 Discussion

### 5.7.1 IGMP Leave Latency Problems and Arguments

As described in Chapter 2, the *IGMP leave latency problem* is one of the most notorious problems of MR-MCC. Some MR-MCC protocols (such as FLID-DL and WEBRC) try to tackle this problem by using *Dynamic Layering (DL)* techniques. As described in Chapter 4, DL avoids layer un-subscription processes (which cause IGMP leave delay) by letting the source reduce the rate of every layer periodically. Hence, to reduce reception rate, the receiver does not need the sluggish un-subscription processes anymore. However, the trade-off is that the receiver has to do more layer subscription processes to maintain and increase its reception rate.

Nevertheless, the DL technique causes more complexity to MR-MCC. Scheduling the layer subscription processes for the DL can be very difficult: if scheduling them too late, data reception may be interrupted, thus causing an unsmooth reception quality for multimedia; if scheduling them too early, unwanted data from the next layer may be received and cause congestion. In some cases, due to the need of more layer subscription processes, the dynamic layering may increase the number of control messages seriously, and cause congestion as discussed in [140].

Due to the aforementioned reasons, we argue that the DL is not a favourable technique to be deployed with MR-MCC. In fact, the IGMP leave latency problem is a problem of IGMP protocol, not MR-MCC protocols. It should be solved at the IGMP level instead. Hence, ERA is neither designed by using DL nor equipped with any techniques to tackle the IGMP leave latency problem.

One way to solve *the leave latency problem* at the IGMP is deploying the *Fast IGMP* [146] proposed by Rizzo. With the Fast IGMP, the IGMP protocol is enhanced by a *Predicative Technique* that can speed up the multicast group leaving processes. According to [146], the proposed modifications of IGMP to be Fast IGMP are needed on the router-side only, and the Fast IGMP is compatible with the existing standard IGMP. Furthermore, the trend of IGMP's specification towards leave latency problem is getting smaller. The leave latency has reduced from 260 seconds (in IGMP version 1 [47]) to 9 seconds (in IGMP version 2 [58]). During the time of writing this thesis, a new version of IGMP (version 3) [35] has just also been standardised by IETF.

### 5.7.2 Security Considerations

Like other protocols, ERA can be subjected to some security problems. However, it is only a design of the congestion control mechanism, and does not aim at providing any special security mechanisms. In deploying ERA, some security mechanisms must be considered together with congestion control mechanisms. In this section, we discuss the security concerns of ERA as follows:

**Denial-Of-Service (DoS) Attacks using Forged Packets**

According to [12], a *Denial of Service (DoS) Attack* is *"a type of security breach to a computer system where users are deprived of the services that they would normally expect to have. For example, a website accessed by millions of people can occasionally be forced to temporarily cease operation. A denial of service attack can also destroy programming and files in a computer system. Although usually intentional and malicious, a denial of service attack can sometimes happen accidentally. "*

*DoS* attackers may try to confuse ERA congestion control mechanism by sending *forged packets* to the session. The forged packets then would prevent successful reconstruction, or cause inaccurate reconstruction of large portions of an object by receivers. This security attack can mainly affect network elements and receivers downstream of the attack. According to [105], the solutions may be: (1) Enable *Reverse Path Forwarding* checks at all routers along the path from the sender to receivers – thus preventing bad agents injecting forged packet into multicast tree data path; (2) Detect the forged packets by using *Packet Authentication Protocols*, such as TESLA [131] and other protocols defined in *IPSEC* [7]. However, the authentication

scheme must offer an instant packet authentication on reception. Otherwise, ERA may suffer severe performance degradation as the delay of authentication may cause ERA to respond to packet reception late. The current authentication technologies have not yet been able to provide such instant authentication, and it is still an open research issue to provide such an instant authentication technique.

**Denial-Of-Service (DoS) Attacks using Corrupted Session Description**

Another way to attack ERA is sending corrupted *SAMs* to confuse receivers. When the receivers obtain a wrong session description (maybe from the fake source), they may not be able to receive the session content correctly. Or, they can be allured to take reception rate at much higher rate than they can, thus causing network congestion. Finally, this can disrupt the network and cause the *DoS*. A possible preventive measurement for this attack is using source authentication techniques to ensure that the session description is really from an authorised source. The authentication protocols standardised in IPSEC can be used.

**Self-Beneficial Attacks**

Unlike *DoS* attackers, *Self-Beneficial Attackers* do not aim at disrupting the network. Primarily, they intend to increase their own bandwidth consumption. To avoid detection (thus preserving the selfish bandwidth consumption), self-beneficial attackers generally try to be low-profiled. Hence, it is more difficult to detect this kind of attack than the *DoS* attack. Self-beneficial attacks may affect: (1) the reception rates of other receivers in the same session with the attacker, (2) the health of the network in the path between the sender and the attacker. Some receivers can have an incorrect or corrupted implementation of ERA to exploit bandwidth share. Due to the

widespread open-source *Operating System*, ERA codes can be easily modified with an evil intention to steal other people's bandwidth. To protect the system from these attacks, receivers may be required to identify themselves as legitimate before they can join an ERA session. The way to do this is still an open-research issue, and is beyond the scope of this research.

**Confidentiality**

*Confidentiality* (or *Privacy*) means that only the intended receivers can decode the data packets. Some data distributed via ERA may be more sensitive to confidentiality concerns. Such confidentiality of the content can be accomplished by *Encryption*. ERA relies on the IP multicast (at network layer), which provides no measure of confidentiality. Hence, to provide confidentiality in this case, we need cryptographic techniques (such as *Data Encryption Standard (DES)*, *Advance Encryption Standard (AES),* and *Rivest Shamir Adleman (RSA) Public Key* [13]) to be used with ERA.

**Integrity**

To ensure the integrity of a received object before delivery to an application on top of ERA, an *Integrity Check Technique* (such as Message Digestive 5 (MD5) hash [142]) may be made on the received object. In addition, to obtain strong cryptographic integrity protection, a *Digital Signature* verifiable by the receiver should be computed on top of the hash value.

Potentially, there would be also other possible security problems/concerns about ERA apart from those mentioned above. Like other proposed protocols, the security issues need a rigorous further review before implementing in the real networks. However, it

is not the purpose of this research to define a complete outline of security issues for multicast protocol.

### 5.7.3 Packet-pair Arguments

Available bandwidth estimation has been first proposed by Keshav using *Packet-pair Probe* of Keshav [86] since 1991. Even though the PP idea is very simple, and works effectively in simulation environments, it is quite challenging to get accurate available bandwidth estimation from the PP in practice. The real world is full of noises and fluctuation of cross-traffic that can lead to wrong estimation [129]. Legout et al. [95] have also revealed that in the environment with various packet sizes, the PP may misestimate the available bandwidth. Hence, it is still an open research to improve the techniques to estimate available bandwidth.

Nevertheless, we believe that available bandwidth estimation techniques inherited from PP are promising solutions. In simulation environments, PP has proved to be very effective, at least until the congestion became so great that probe packets become regularly lost. This effectiveness has been evidenced by our experiments in Chapter 6 and other research work, such as [114], [129], and [160]. Furthermore, many studies have recently improved the accuracy of PP techniques, such as the *receiver-side Packet-pair bunches Probe* of Paxson [129] (which is also used in our design) and *Packet Trains* ([17], [54], and [114]). Hence, the accuracy of bandwidth estimation by PP-based technique has a tendency to improve.

Moreover, PP does not exist only in simulation-based work, but also in real-life implementations. There have been several proposals relying on PP for available

bandwidth estimation in the real-life networks, such as *PathRate* by Dovrolis et al. [55], *NettiMeter* by Lei [90], *Network Diagnostic Tool (NDT)* by Carlson [37], *Swift Start Proposal* by Sterbenz et al. [154], *Initial Gap Increasing (IGI)* by Hu et al. [72], and *PeriScope* by Harfoush et al. [69], [70]. All these projects have provided modules for kernel implementation, which are good examples of PP-based available bandwidth estimation techniques on the real network. Consequently, we believe in available bandwidth estimation techniques and have designed ERA by using them. Any improved techniques of PP in the future would be able to be applied into ERA and replace the technique used in Section 5.5.5.

For the severe congested situation, this is indeed a pathological case for PP technique. In this situation, the probe packets become regularly lost. Hence, relying on the PP alone would certainly cause a problem. Yet, ERA has been designed to avoid this problem. As mentioned in Section 5.5.2, when the packet loss rate becomes high, ERA will not rely on PP available bandwidth estimation but use TCP-friendly rate estimation instead. Hence, ERA does not suffer from this pathological situation.

## 5.8 Summary

In this chapter, we have addressed the design aspect of ERA, a new MR-MCC protocol. ERA design is done using explicit rate notification based on the receiver-sided Packet-bunches probe and the TCP equation. The goal of ERA is to provide: scalability, responsiveness, fast convergence, low packet loss rate, and fairness including TCP-friendliness. We have also successfully implemented the design in ns-2. The implementation arguments of ERA have also been discussed in this chapter. The performance evaluation of ERA will be presented in the next chapter.

# Chapter 6

# ERA: Performance Evaluation

## 6.1 Introduction

In the previous chapter, we have proposed the design and implementation of a new experimental MR-MCC protocol named ERA. This chapter aims at testing the performance of ERA using simulation experiments. Furthermore, we compare ERA with other MR-MCC proposals. To do so, we have modelled some simulation scenarios, and run several experiments to study it. The evaluation criteria are responsiveness, efficiency of network utilisation, inter-protocol fairness (especially TCP friendliness), intra-protocol fairness, intra-session fairness, packet loss ratio, feasibility and scalability. Details of these criteria have been given in Chapter 4.

The remainder of this chapter is organised as follows. In Section 6.2, we describe simulation tools. Section 6.3 explains parameter setting. Performance metrics are provided in Section 6.4. The simulation scenarios, objectives and results are presented in Section 6.5. Section 6.6 provides performance comparison of ERA with other MR-MCC protocols. Finally, a summary is given in Section 6.7.

## 6.2 Simulation Tools

The ns-2 network simulation package [121] is used for our experiments. We have integrated ERA algorithms into ns-2 version 2.1b6a. The details of implementation and validation can be found in Chapter 5. The overview about the network simulation technique used in this chapter has been explained in Chapter 3. The oTcl simulation scripts used for our experiments in this chapter are available at the author's *Research Log Web Page* [136].

## 6.3 Simulation Parameters

In this section, we define the default parameters used in our experiments. The packet size of all flows (ERA and TCP) is chosen to be 512 bytes. The router's queuing scheme is drop-tail, with a queue size of twice delay-bandwidth product.

| Parameters | Default Values |
|---|---|
| Layering Scheme | Equal |
| Layer granularity | 20 Kbps |
| Number of layers | 100 |
| Length of PP bunch | 2 |
| Min PP required | 3 |
| Rate Adaptation Interval | 1 second |

**Table 6-1 ERA Default Parameters**

ERA's default parameters are summarised in Table 6-1. The layer organisation for ERA is *equal scheme*. Each layer has the same size of granularity of 20 Kbps, and the number of layers is set to 100. The length of *Packet Pair Bunch* is set to two for our simulation. In a real implementation, a bigger length could be used to gain a more accurate estimation of available bandwidth. The minimum number of PPs required to estimate available bandwidth is set to three. If there are fewer than three packet-pairs received, ERA will assume that the packet pairs are lost during severe congestion. In this case, it cannot make a good estimation of available bandwidth using PP. According to its rate adaptation algorithms, ERA uses the TCP-throughput equation to calculate the target rate instead. The full details of ERA's rate adaptation algorithms have been discussed in Chapter 5.

In our experiments, the *Rate Adaptation Interval (RAI)* is set to one second. This means ERA will adjust its target rate every second. This value is arbitrarily set only for our simulation purpose. In a real implementation, *RAI* would be set according to the requirement of applications running on top ERA. Some applications may prefer a short *RAI* to be very responsive to the network conditions, while some applications (such as multimedia applications) may prefer a longer *RAI* to maintain smoothness of reception quality. It is not the purpose of this thesis to discover the best *RAI* value of different multicast applications; optimal *RAI* setting is left as future work.

For the experiments that also simulate TCP connections, we use the ns-2 implementation of TCP Reno. The maximum size of TCP congestion window is set to 2000 packets to remove the effect of the maximum window size. The applications on top of TCP sources are infinite FTP sessions that have unlimited data to send.

Each simulation is run 20 times using different RNG seeds. Results are averaged and quoted with respect to confidence intervals of 95%. The error bar is shown where it is appropriate. We do not show error bars when intervals are very small or negligible. Further details of parameter settings, sensitivity test, and confidence intervals have been given in Chapter 3.

## 6.4 Performance Metrics

Throughput, efficiency ($E$), convergence time, TCP friendliness ratio ($F$), subscription level, and PLR are the metrics used to evaluate ERA. A detailed description of each metric has been given in Chapter 4.

## 6.5 Simulation Experiment

In this section, we discuss simulation scenarios, objectives, and results of five experiment sets used to evaluate ERA. We start from Experiment I testing how fast and with what stability ERA converges to an optimal rate. Experiment II tests how ERA responds to the changes of available bandwidth. Then, the intra-protocol fairness of ERA towards TCP is tested in Experiment III. Experiment IV tests intra-protocol fairness. Finally, the coordination of downstream receivers and intra-session fairness of ERA are evaluated in Experiment V.

### 6.5.1 Experiment I: Convergence to Target Rate

#### Simulation Scenario and Objectives

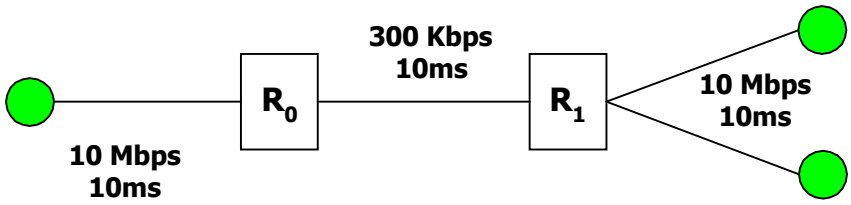



**Figure 6-1: Simulation Topology of ERA Experiment I**

The objective of this experiment is to test how ERA converges to the target rate, and to verify that ERA receivers can estimate the available bandwidth properly and adjust the subscription level to the optimal level quickly. We use the topology depicted in Figure 6-1 of a single multicast source with two receivers. The input bandwidth is 10 Mbps, while the bottleneck link is 300 Kbps. We start the multicast source at time zero and its sinks randomly three seconds later. The simulation is run for 80 seconds.

**Simulation Results and Discussion**

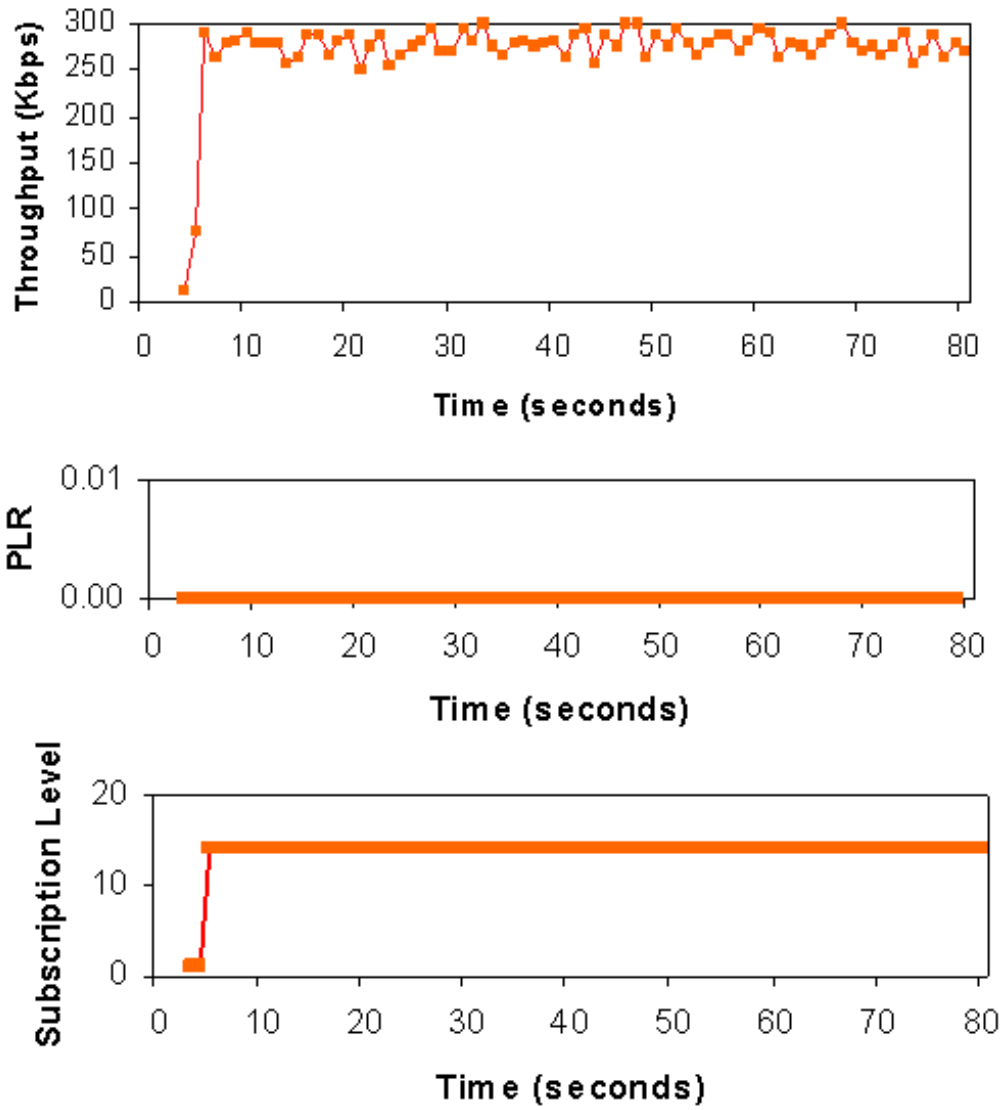



**Figure 6-2: Convergence to target rate**

Figure 6-2 shows the results of this experiment. From the figure, we can see that ERA is very fast to converge. The convergence time is only 2 seconds for ERA to converge to subscribe 15 layers without causing packet loss. It is also highly efficient in utilising the available bandwidth. The average throughput gained is approximately 271.93 ± 1.42 Kbps. The efficiency of network utilisation (*E)* is 0.9 ± 0.03.

This is because ERA uses PP to estimate the available bandwidth. So, it can decide the optimal reception rate without over-using the bandwidth and causing packet loss. Its rate adaptation algorithms also allow it to converge quickly to the optimal level.

### 6.5.2 Experiment II: Response to Network Conditions

**Simulation Scenario and Objectives**

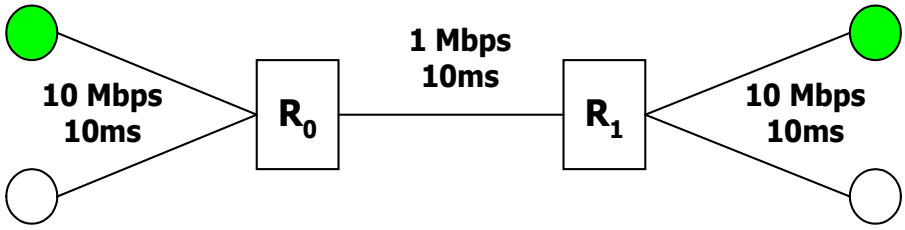



**Figure 6-3: Simulation Topology of ERA Experiment II**

The objective of this experiment set is to test our protocol in terms of responsiveness, packet loss ratio, efficiency of network utilisation and smoothness when the available bandwidth changes during a transmission. We use the topology depicted in Figure 6-3 of a single multicast session sharing with a Constant Bit Rate (CBR) session. The input bandwidth is 10 Mbps, while the bottleneck link is 1 Mbps. We start the multicast source at time 0 and its sink 3 seconds later. At time 20 seconds, we start a CBR source sharing over the bottleneck link at rate 500 Kbps to use half of the bottleneck bandwidth and see how ERA adjusts to the change of available bandwidth.

**Simulation Results and Discussion**

Figure 6-4 shows the graphs of throughput, PLR and subscription level against the simulation time. From the results, ERA can react rapidly to the change of the available bandwidth and show very fast convergence, responsiveness, and smooth throughput.

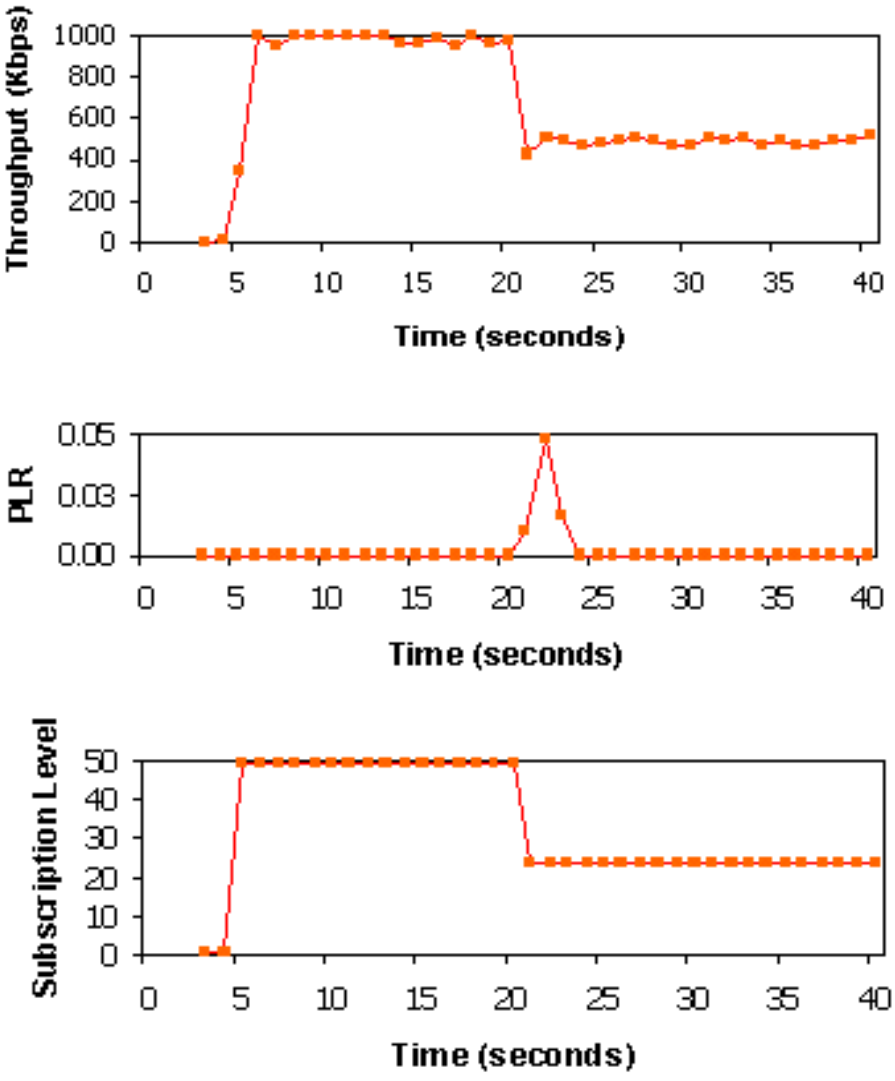



**Figure 6-4: Response to changes in available bandwidth**

From the figure, when ERA starts after 3 seconds, it takes roughly 3 seconds converging to 50 layers to have the optimal rate around 1 Mbps. It also shows responsiveness to the changes of network condition when the available bandwidth is

halved after 20 seconds. ERA takes only 1 second to adjust its reception rate to suit the available bandwidth. The figure also shows very low packet loss rate of the whole simulation (0-0.05 only).

The low packet loss rate and efficient bandwidth utilisation result from that by using its available bandwidth estimation, ERA can sense the changes of network conditions. The rate adaptation of ERA then tries to adjust the reception rate not only to avoid over-using available bandwidth but also to efficiently utilise it.

### 6.5.3 Experiment III: Bandwidth Share with TCP

**Simulation Scenario and Objectives**

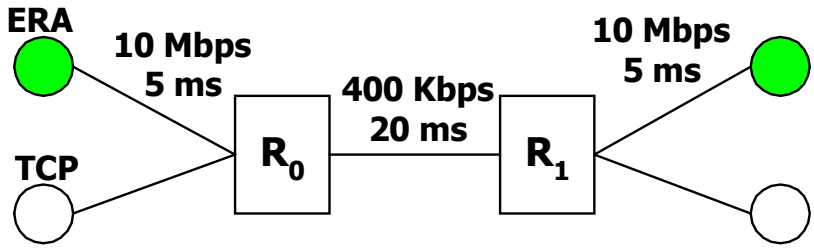



**Figure 6-5: Simulation Topology of ERA Experiment III**

In the previous experiment set, we have seen that ERA congestion control mechanisms can respond to congestion very well. However, just being responsive to congestion is not enough. It also needs to be responsive at the same level as TCP in order to maintain TCP friendliness. So, this further experiment set aims at investigating the behaviour of ERA when sharing bandwidth with TCP.

We deploy the dumbbell topology depicted in Figure 6-5, using a single multicast session sharing with a TCP session. We start one session at the beginning of the simulation and another at time 20 seconds. This is to inspect two cases: (1) when TCP starts first, and (2) when ERA starts first. Each exterior link's bandwidth is 10 Mbps, while the bottleneck link is 400 Kbps. The delay of each exterior link and the bottleneck link is set to 5 and 20 milliseconds, consecutively. We start the multicast source at time zero and its sink randomly three seconds later. The simulation is run for 200 seconds.

**Simulation Results and Discussion**

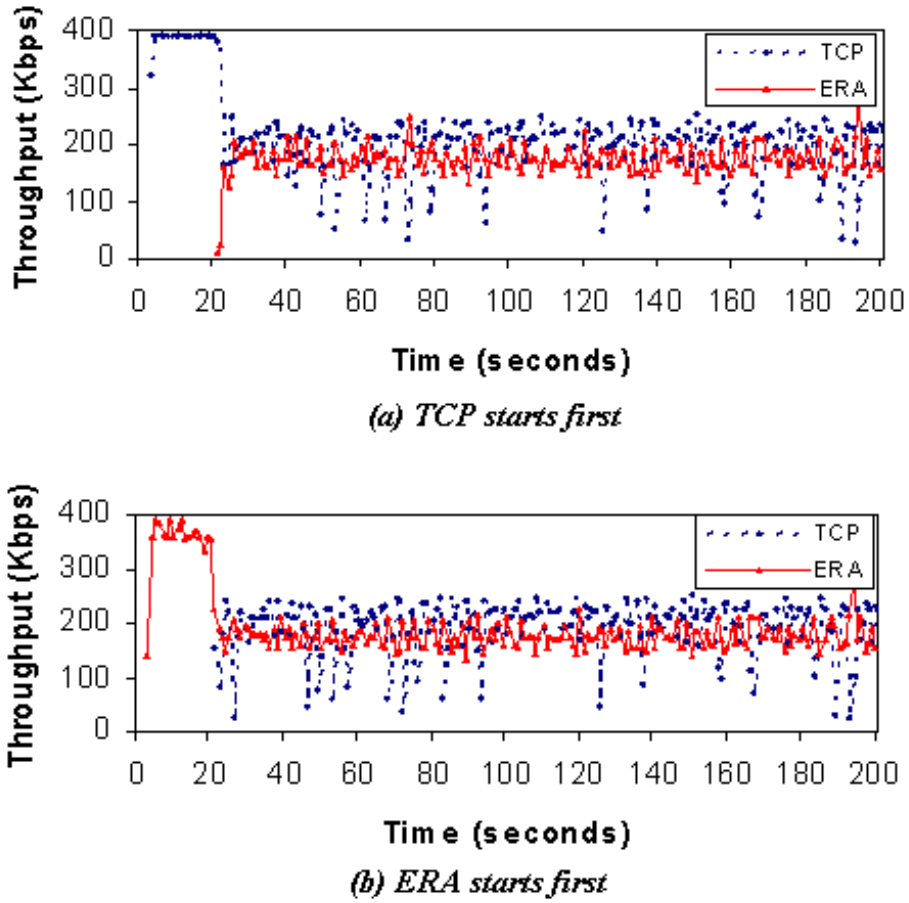



**Figure 6-6: Bandwidth share with TCP**

From Figure 6-6, the results show good TCP friendliness of ERA regardless of whether TCP or ERA started first. The figure shows very clearly that even when ERA starts first, it does not starve the competing TCP connection. Furthermore, it does not let the TCP connection starve itself, when TCP starts first.

From the figure, after 20 seconds when competing with TCP on the same bottleneck link, ERA and TCP can fairly share bandwidth (around 200 Kbps each). The average throughput during the last 100 seconds of ERA and TCP is approximately $196.4 \pm 1.7$ Kbps and $176 \pm 0.7$ Kbps, consecutively, with $F$ approximately equal to 0.9 whether TCP or ERA starts first.

This is because ERA's rate adaptation algorithms have included the TCP-throughput equation to calculate the rate gained by TCP on the same network conditions as its target rate.

We note that being fair to TCP does not mean always getting exactly the same throughput with TCP. Even TCP has not been fair to itself because the throughput is inversely proportional to RTT, varying due to the queuing delay.

The ratio of TCP traffic in the Internet at the moment is 90%, as mentioned previously. However, our simulation scenarios use the traffic composition ratio between TCP and ERA of 50:50. To evaluate TCP-friendliness of ERA in the environment of 90:10 traffic composition ratio, we rerun our simulation using the same scenario (depicted in Figure 6-5), but with 9 TCP and 1 ERA connections. Each exterior link's bandwidth is 10 Mbps, while the bottleneck link is 1 Mbps. The results are shown in Figure 6-7, and demonstrate TCP-friendliness of ERA.

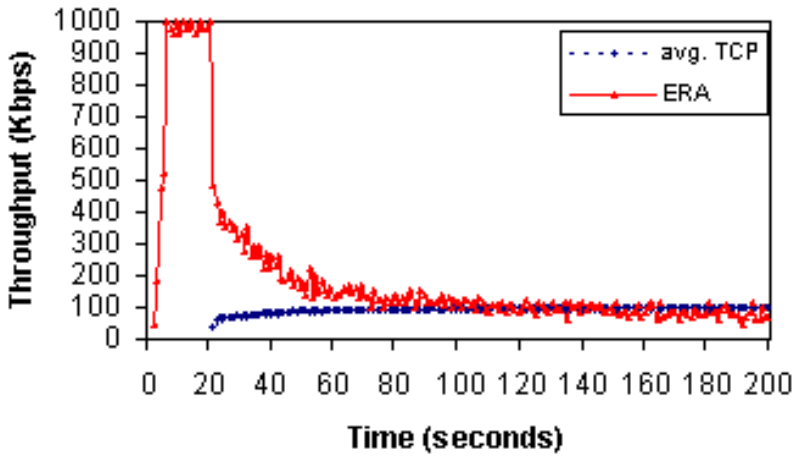



**Figure 6-7: Bandwidth share with TCP in 90:10 traffic composition ratio**

### 6.5.4 Experiment IV: Intra-protocol Fairness Test

**Simulation Scenario and Objectives**

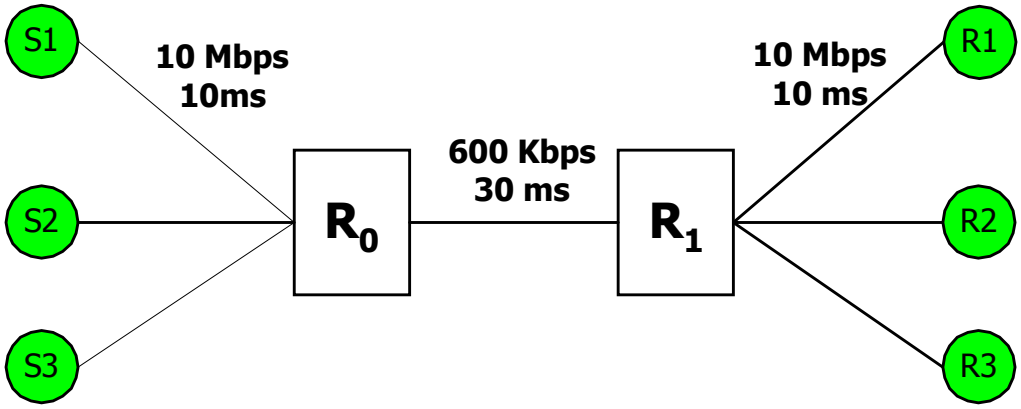



**Figure 6-8: Topology of ERA Experiment IV**

The objective of this experiment set is to test the intra-protocol fairness of ERA. The topology depicted in Figure 6-8 is used with three multicast sessions. Each one consists of one source and one sink. The bottleneck bandwidth is 600 Kbps, while each exterior link bandwidth is 10 Mbps. The simulation is run for 100 seconds. We start the first two sessions randomly during the first ten seconds, and start the third session at time 50 seconds.

**Simulation Results and Discussion**

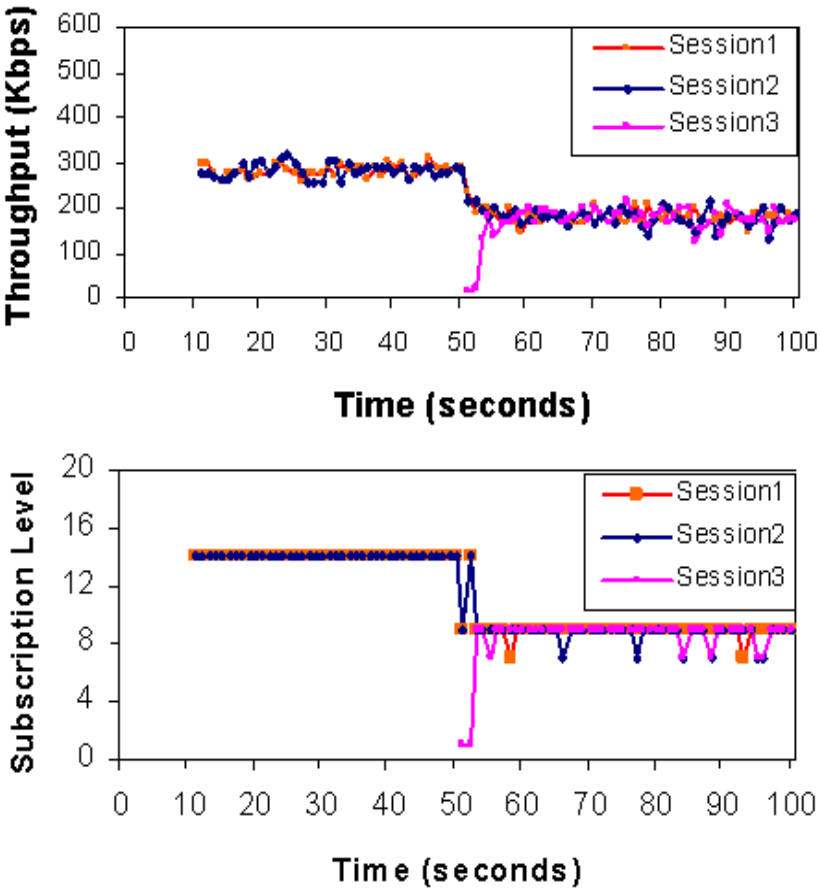



**Figure 6-9: Intra-protocol fairness of 3 sessions**

From Figure 6-9, we can see that ERA demonstrates its intra-protocol fairness. When the first two sessions (Session 1 and 2) start, they can share the bottleneck bandwidth fairly (around 300 Kbps each) by subscribing 15 layers. After 50 seconds, when another ERA session (Session 3) starts, three sessions of ERA can still adjust the rate to be fair to each other to around 200 Kbps by each subscribing 10 layers.

### 6.5.5 Experiment V: Co-ordination of Receivers

**Simulation Scenario and Objectives**

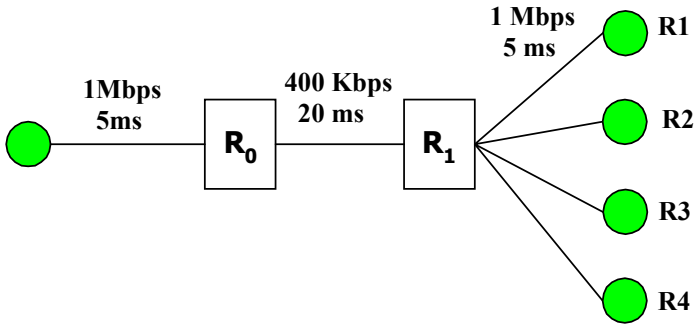



**Figure 6-10: Topology of ERA Experiment V**

As mentioned in Chapter 2, the coordination among downstream receivers of MR-MCC protocol is very important. Uncoordinated subscription can cause an inefficiency of bandwidth utilisation, and unfairness among the downstream receivers (called intra-session unfairness). Uncoordinated un-subscription can also make the congestion problem persist. Hence, the objective of this experiment set is to test the coordination of downstream receivers behind the same router, the intra-session fairness, and the effect of late joiners.

We use the topology depicted in Figure 6-10, using a single multicast source with 4 receivers. The bottleneck bandwidth is 400 Kbps, while each exterior link bandwidth is 1 Mbps. The delay of each exterior link and the bottleneck bandwidth is 5 milliseconds and 20 milliseconds, consecutively. We start receiver R1 and R2 at time 3 seconds, and receiver R3 & R4 (as late joiners) at time 20 and 40 seconds, consecutively.

**Simulation Results and Discussion**

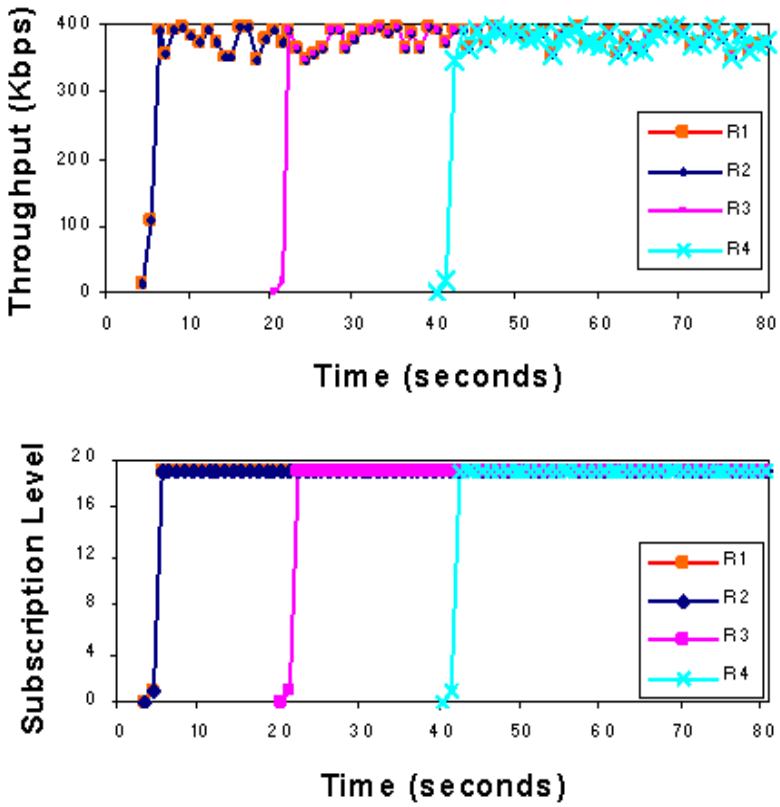



**Figure 6-11: Co-ordination of receivers and late joiners**

Figure 6-11 shows a very good intra-session fairness and coordination among the downstream receivers of ERA. Furthermore, the late joining of extra receivers is not a problem for ERA. All late joiners (R3 and R4) can synchronise very well with the previous receivers under the same bottleneck link. All receivers can converge very fast and gain optimal bandwidth consumption.

When R1 and R3 start after 3 seconds, both adjust the subscription level to 20 layers and utilise bandwidth around 400 Kbps. After 20 and 40 seconds, the late joiners R3 and R4 can also adjust their subscription level and reception rate to be the same as R1 and R2. In summary, all downstream receivers, including two later joiners of the same

transmission session, demonstrate a very good co-ordination and intra-session fairness to each other.

## 6.6 Comparison of ERA with other MR-MCCs

In this section, we compare the performance of ERA with other MR-MCC protocols. RLM, RLC, ALMA, CIFL, FLID-DL, MLDA, PLM, and WEBRC are the MR-MCC proposals used for comparison. An overview of these protocols can be found in Chapters 2 and 4.

For comparing these protocols with ours, we analyse their designs and the performance results reported by their authors. Furthermore, we choose some representative evaluation cases as explained in the papers presenting these protocols. We then re-simulate those cases in our simulation environment using ns-2. The results achieved by ERA are compared with the results of the other protocols reported by their authors. This is to reduce possible errors in the comparisons due to misinterpretations or wrong implementations of the protocols. In addition, since some protocols have not been implemented in ns-2 and some of them have no ns-2 modules available publicly, we cannot re-simulate all of them.

The evaluation criteria are responsiveness, network utilisation, packet loss rate, TCP friendliness, scalability and feasibility. The details of these evaluation criteria have been discussed in Chapter 4.

### 6.6.1 Comparison with RLM and RLC

From the literature ([34], [88], [113], [140], [159]), mentioned in Chapter 2, there are several fundamental problems of RLM and RLC reported, such as slow convergence, unresponsiveness, TCP-unfriendliness, and high packet losses. This is because they have been designed using the join experiment. For RLC, although it is enhanced with the *Burst Test technique*, it is still prone to over-subscription and high packet losses and slow convergence. Furthermore, RLC was designed to be fair towards TCP whose RTT is one second only. It is aggressive towards TCP with RTT larger than one second, and submissive towards TCP with RTT smaller than one second. Our experimental results of ERA previously in this chapter have shown their superiority over RLM and RLC in terms of responsiveness, the efficiency of network utilisation, packet loss rate and TCP-friendliness.

### 6.6.2 Comparison with FLID-DL

We have extensively investigated FLID-DL in Chapter 4 and shown that FLID-DL is not TCP-friendly, slowly convergent, and has relatively high packet loss rate. Compared with the experimental results of ERA in this chapter, we can see that ERA performs better.

### 6.6.3 Comparison with PLM

Both PLM and ERA rely on PP to infer the available bandwidth. So, in terms of responsiveness, network utilisation and PLR, they could gain similar performance. However, PLM requires FQ at every router to enforce TCP-friendliness. This requirement is unfeasible, as some routers on the Internet still provide only drop-tail FIFO queuing. From our experiment in Chapter 4, Section 4.6.6, we have shown that

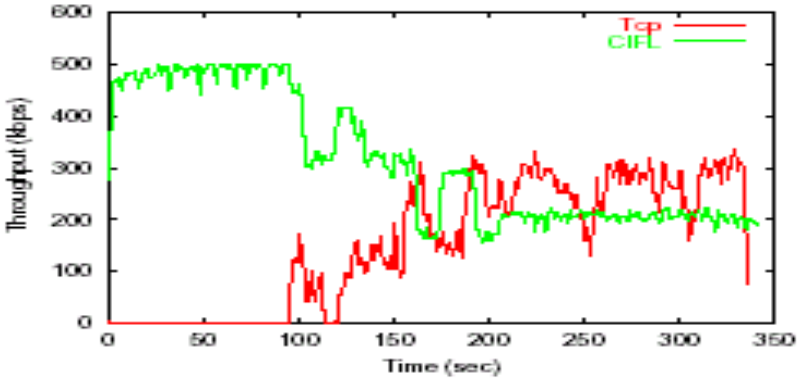
PLM without FQ can cause starvation of competing TCP connections. Hence, in terms of TCP friendliness, ERA provides more feasible algorithms to achieve this goal.

### 6.6.4 Comparison with ALMA

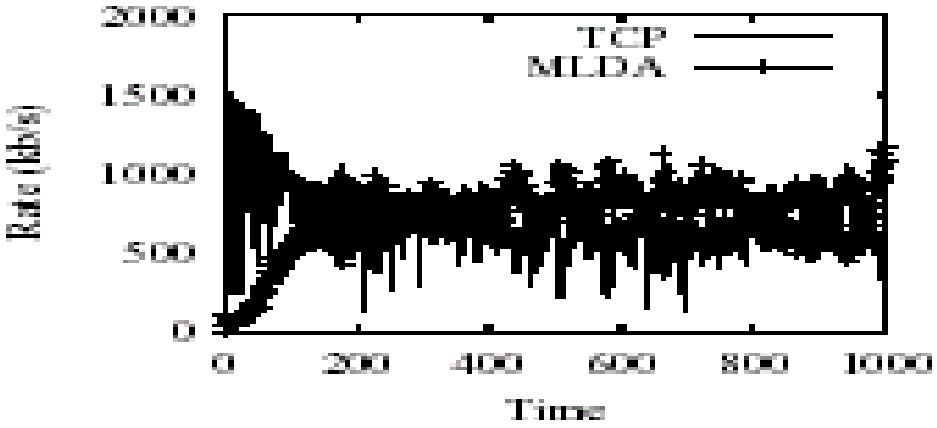
Comparing ALMA with ERA, there are two disadvantages of its design – unfeasibility and TCP-unfriendliness. ALMA is unfeasible because it relies on the active networks paradigm, which is still far from the deployment stage. In terms of TCP-friendliness, ALMA is not designed to be TCP-friendly, as confessed by its author [168]. Sari et al. [148] have also shown the aggressive behaviour of ALMA towards TCP. From their experiments, ALMA can cause starvation of competing TCP connections. Hence, the deployment of ALMA could be very dangerous to TCP traffic, which is the majority of traffic on the Internet.

### 6.6.5 Comparison with CIFL, MLDA and WEBRC

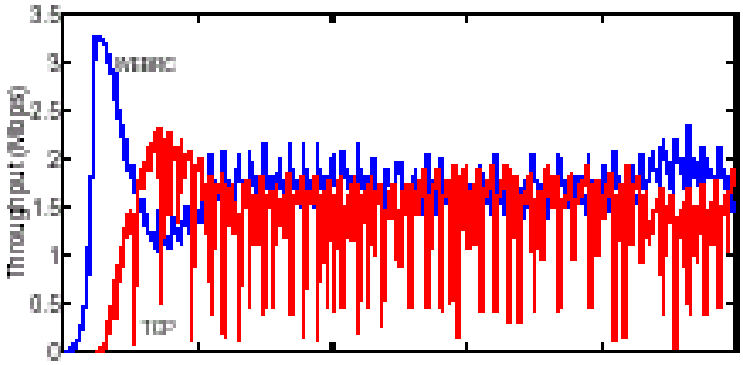
In terms of TCP-friendliness, CIFL, MLDA, WEBRC and ERA have been designed by integrating the *TCP Throughput Equation* into their rate adaptation scheme. Hence, they have comparable TCP-friendliness, as shown in Figure 6-12.




**(a) CIFL** *(Source: [88])*




**(b) MLDA** ***(Source: [151])***




**(c) WEBRC** *(Source: [102])*

**Figure 6-12: TCP-friendliness of CIFL, MLDA and WEBRC[6]**

[6] Apart from these figures, we cannot show raw data of results for each protocol, as they have not been provided in the original paper. However, the figures roughly show TCP-friendliness of each protocol.

However, for CIFL, we expect it would have a higher PLR than ERA due to relying on a variant of the join experiment. In contrast, ERA uses PP to detect available bandwidth and tries to avoid network over-use before causing any packet loss. Furthermore, CIFL is designed using the old TCP-throughput equation [107], which is reasonable only at low PLR (below 0.16), as demonstrated in [124]. When the PLR gets higher (above 0.16), we expect problems with CIFL in terms of TCP-friendliness. However, we cannot yet demonstrate this via network simulation, since the ns-2 modules of CIFL are not available publicly.

In terms of scalability, MLDA would under-perform compared with other MR-MCC protocols, including ERA. This is because MLDA proposes to have feedback from receivers to report the rate to the sender. Then, the sender can reduce the rate on a layer, which causes congestion. On one hand, this makes MLDA react quicker to the congestion than just waiting for all receivers to unsubscribe from that layer. However, this causes less scalability and more complexity of the sender and application to distribute the data on such a dynamic layering scheme. We argue that the key purpose of MR-MCC is to provide a scalable solution for multicast congestion control. So, any MR-MCC implementation schemes that cause severe scalability degradation would not be a good choice.

WEBRC also maintains the join-experiment-like concept, so we expect higher PLR than with ERA. To demonstrate this, we simulate ERA using the same scenario as a test case of WEBRC in Section 5.1.2 in [102].

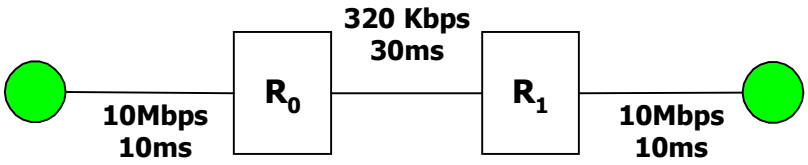



**Figure 6-13: Simulation Topology: WEBRC vs. ERA**

The simulation topology is shown in Figure 6-13. We use only one source and one sink. The link between two routers is the bottleneck link, with 320 Kbps of bandwidth and 30 milliseconds of delay.

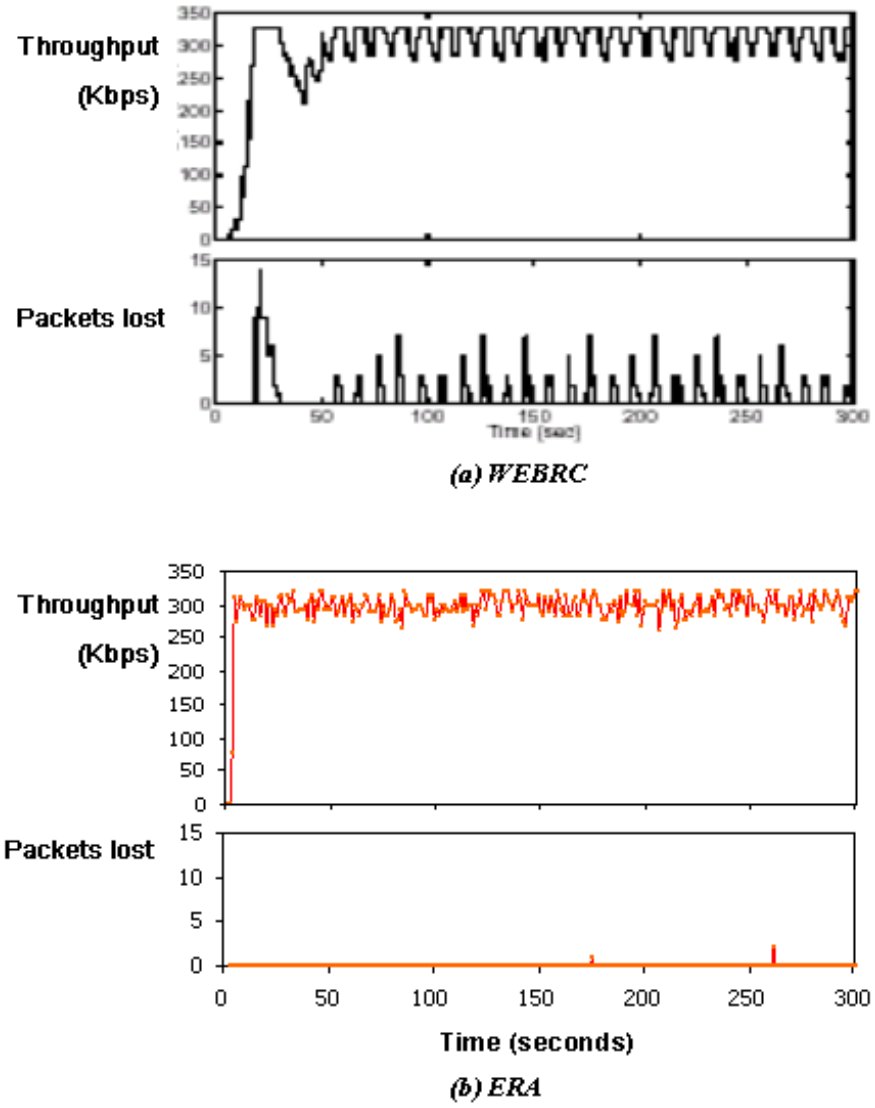



**Figure 6-14: Convergence to optimal rate (WEBRC vs. ERA)**

**(a) WEBRC** *(Source [102])* **(b) ERA**

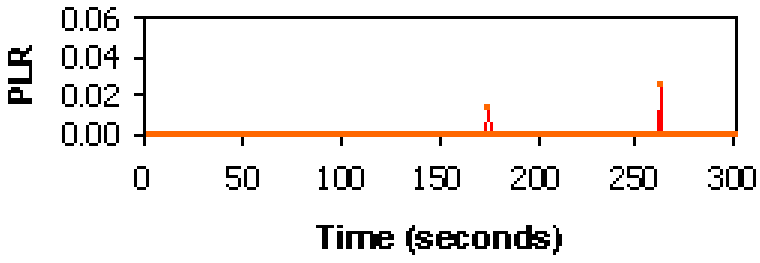



**Figure 6-15: Packet Loss Ratio of ERA**

The results are shown in Figure 6-14 and 6-15. From the results, both ERA and WEBRC have good performance. The results show fast convergence and low PLR. However, ERA has shown less packet loss than WEBRC. The PLR is at 0 almost all the simulation time for ERA, with the worst PLR at 0.03 only. This is because the PP used by ERA is better in handling congestion avoidance. With PP, ERA can estimate the available bandwidth and try not to over-utilise it. Furthermore, from the graph, we can see that ERA is faster to converge to the optimal rate.

In addition, WEBRC algorithms are more complicated compared to ERA. This is because of the wave-like scheme used by WEBRC to avoid the IGMP leave latency problem. Indeed, IGMP leave latency is a big problem of layered multicast congestion control protocols. However, we argue that this problem is a problem of the IGMP protocol and should be solved at the IGMP, not at the congestion control protocol. The details of the IGMP leave latency problem and the possible way to solve it at IGMP protocol have been discussed in Chapter 5.

To summarise, the comparison of ERA with several MR-MCC protocols is shown in Table 6-2.

| MR-MCC protocols | Disadvantages in comparison with ERA |
|---|---|
| RLM | (1) Unfairness including TCP-unfriendliness<br>(2) High PLR due to relying on *join experiment*<br>(3) Slow convergence<br>(4) Less scalable due to use of synchronised messages between receivers |
| RLC | (1) Designed to be fair to TCP only with RTT = 1 second<br>(2) Slow convergence |
| ALMA | (1) Unfeasibility due to relying on Active Networks paradigm, which is still far from deployment stage<br>(2) TCP-unfriendliness (aggressive towards TCP) |
| CIFL | (1) Possibility of having higher packet loss due to relying on a variant of join experiment<br>(2) Relying on old TCP-throughput equation, which is reasonable only at low PLR (below 0.16) |
| FLID-DL | (1) High PLR due to relying on the join experiment<br>(2) Slow convergence<br>(3) TCP-unfriendliness |
| MLDA | (1) Less scalable due to the use of feedback from receivers to sender<br>(2) More complexity of protocol and applications since they have to distribute data onto dynamic layers |
| PLM | (1) Unfeasibility due to relying on FQ to be existed at all routers<br>(2) TCP-unfriendliness in environment of drop-tailed FIFO queuing (even causing TCP starvation) |
| WEBRC | (1) Higher PLR<br>(2) Algorithms more complicated due to dynamic layering, which may not be necessary. |

**Table 6-2: Performance comparison ERA with other MR-MCCs**

## 6.7 Summary

In this chapter, we have done simulation experiments to study the performance of ERA. The experiments show that ERA can provide: responsiveness, fast convergence, efficient network utilisation, low packet loss rate, TCP friendliness, intra-protocol fairness, and intra-session fairness. We have also discussed the performance comparison of ERA with other MR-MCC protocols (RLM, RLC, ALMA, CIFL, FLID-DL, MLDA, PLM and WEBRC).

# Chapter 7

# Conclusions and Future Work

## 7.1 Summary and Discussion

Multicast congestion control is a very significant issue to be conquered to allow the deployment of multicast applications on the Internet. To achieve the scalability of multicast congestion control for a very large heterogeneous group of receivers, the multi-rate multicast congestion control (using receiver-driven layered multicast techniques) has been accepted by the research community as a promising solution.

In this thesis, we have presented our work on the investigation, simulation, and evaluation of earlier proposed MR-MCC protocols. Then, the lessons learnt from the performance evaluation are used to design, model, implement, simulate and evaluate our new experimental MR-MCC protocol.

In Chapter 3, we presented the methodology used in this research. Three performance evaluation techniques (analytical modelling, simulation and measurement) have been discussed and compared. We have found that the cost of prototyping, building a test-bed and measuring on MR-MCC protocols would be prohibitive and beyond our reach. Also, analytical modelling is too limited (by its simplification and assumptions) to be used as an efficient tool to model an MR-MCC protocol. We have therefore selected network simulation as our method.

While network simulation is a useful and vital method for telecommunication study, during the time of this research, we have learnt that to perform an accurate, trustable simulation study is not trivial. There are several possible pitfalls of using network simulation, such as unclear objectives, unsuitable scenarios, bad performance metrics, insensible parameter settings, lack of parameter sensitivity analyses, lack of statistic analyses and bad presentation of results. According to the survey of Pawlikowski et al. [127], more than 70% of network-simulation-based papers were not conducted with a strong and valid simulation technique. Hence, in this research, we have carefully constructed a rigorous simulation study by defining clear objectives for each simulation, modelling and justifying simulation model and parameters (including parameter sensitivity testing), carefully implementing simulation scripts and protocol codes, validating and verifying our tools, taking suitable output analysis through statistical tools, and properly presenting our results.

Because of its open source codes, robustness, and wide support, the ns-2 network simulation package has been selected as our simulation tool. Although ns-2 is open-sourced, widely deployed and supported by its users/developers around the world, we

have found that it is not trivial to deal with this network simulator. For example, ns-2 version 2.1b6a is compiled smoothly by gcc.2.9x.x (C++ compiler included with old versions of Linux). However, for gcc 3.x.x, included with Redhat version 7 (up), there are several compile errors that require us to correct the codes. The experience from the compilation difficulties has been logged and made available to be useful to others at [136]. Furthermore, extended modules of ns-2, which are not included in the simulation package by default (such as FLID-DL and PLM used in our experiments) can cause problems in compiling, validating and integrating with the ns-2 main package. In addition, to implement a new protocol in ns-2 is not simple; especially the use of oTcl linkage of ns-2 can be cryptic when started.

In Chapter 4, we have focused on studying four key existing MR-MCC protocols – RLM and RLC (the archetype of MR-MCC protocols), and two other recently proposed protocols – FLID-DL and PLM (extended and improved MR-MCC protocols from the first two). The criteria to evaluate MR-MCC have been established, including responsiveness, efficiency of network utilisation, fairness (including intra and inter-protocol fairness, TCP-friendliness and intra-session fairness), packet loss rate, scalability, speed of convergence, smoothness and feasibility. From the study, we have discovered that the four previous MR-MCC proposals have some major drawbacks. Although RLM has initiated a good framework of receiver-driven layered multicast, it provides no fairness. It also converges slowly to the optimal subscription level. RLC was not designed to be fair to TCP in general, but is designed to be fair only to TCP with RTT of one second. FLID-DL is also TCP-unfriendly and slow convergent. PLM relies on an unfeasible implementation scheme.

Furthermore, we have discovered from our study that the use of the *Join Experiment* technique (to detect congestion) in RLM, RLC, and FLID-DL is fundamentally risky to over-subscribe, and causes high packet loss rate in these three protocols. We have also learnt that PLM, on the other hand, by using an available-bandwidth estimation can feedback quicker to congestion, and cause less packet loss. However, we have found that the way that PLM enforced fairness by making FQ compulsory for its implementation is not feasible.

Hence, in Chapter 5, we have developed a new experimental MR-MCC protocol (ERA) by combining the following proven techniques:

(1) The receiver-sided *Packet-bunch Probe* available bandwidth estimation of Paxson [129]

(2) The TCP-friendly rate estimation of Padhye [123]

(3) The receiver-driven layered multicast framework of McCane [113]

together with

(1) Our new rate adaptation scheme, which is responsive and TCP-friendly

(2) An innovative framework for the cooperation between the sender and receivers.

Our design goal is to provide MR-MCC with scalability, responsiveness, fast convergence, low packet loss rate, and fairness including TCP-friendliness. Our new experimental MR-MCC protocol has been successfully implemented and validated in the ns-2 network simulator. We have also simulated it (in Chapter 6) with various network conditions, and the results demonstrate that all the good properties cited above are held. Furthermore, we have discussed the IGMP leave latency problem,

which is the most notorious problem of MR-MCC, and argued that this problem should be solved at the level of IGMP protocol, not the level of multicast congestion control protocol. In addition, the performance comparison of ERA with other MR-MCC proposals has been discussed and illustrated.

## 7.2 Achievements in this Research

This research has evaluated the previous proposals of MR-MCC protocols, and proposed a new design of MR-MCC successfully. The major contribution of the research can be described as follows:

(1) By drawing upon the literature, this research has established a set of key performance evaluation criteria and performance metrics to evaluate MR-MCC protocols.

(2) A performance study by network simulation to compare two recently proposed MR-MCC protocols (FLID-DL and PLM) against each other and discussion of them in comparison with their ancestor protocols (RLM and RLC) have been successfully completed.

(3) The ideas learnt from the performance evaluation of previous MR-MCC have been deployed to propose a novel design of an innovative experimental MR-MCC protocol (ERA), which offers scalability, responsiveness, fast convergence, fairness (including TCP friendliness, intra-protocol fairness, and intra-session fairness), efficiency in network utilisation, and feasibility to implement. This new design has been successfully implemented in the ns-2 network simulator and tested through several simulation scenarios.

(4) Finally, a performance comparison between our new experimental MR-MCC protocol and other proposals of MR-MCC protocols has been made, and demonstrated a few advantages of our design.

## 7.3 Future Work

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

### 7.3.1 Complex Simulation Models/Scenarios

Our network simulation scenarios are rather simple, especially in comparison with the huge, heterogeneous, constantly changed Internet. While these simple scenarios are useful to explore behaviours of MR-MCC mechanisms, they are still far away from the study to understand the behaviours of MR-MCC protocols in response to traffic dynamics on a huge scale of the real Internet. Yet, there is no current simulation technology that can simulate networks of that size. Even if the model could be scaled, suitable tools to interpret effectively the results are still difficult to find. Hence, the issues of simulation scale remain one of the simulation issues yet to be tackled. So far, there has been just a little initial research work on this issue, such as [36], [53], [171] and [172]. So, this would be an avenue for future work.

### 7.3.2 Completing the Specification of ERA

Although we have modelled most of the components of ERA and simulated it extensively, it is still only an *experimental protocol*, and far from a *complete protocol*. Some points of its specification are still left as open-research issues. They can be discussed as follows:

**Multicast Round Trip Time Measurement**

As mentioned in Chapter 5, multicast round trip time measurement is still an open-research issue and is leftover in our model. The best technique found in the literature may be the technique proposed by Luby et al.[102]. This technique calculates the multicast round trip time from the difference between the time of issuance of join request and arrival time of the first packet of the layer. However, this is only a rough estimation because the join-request messages propagate back only to the closest router to the sender (not the sender). The latency between the sender and the closest router has been missed out. So, multicast round trip time measurement would be a point for future work.

**Security Considerations**

We have discussed possible security issues related with ERA in Chapter 5. In order to move ERA from an experimental protocol to a real implemented protocol, a rigorous review of security issues would need to be done properly. Security considerations of multicast protocols are still at an infant stage, and much work still needs to be done. Hence, *IETF* and *IRTF*[7] have charted the *Multicast Security (MSEC)* [5] and the

[7] Internet Research Task Force (http://www.irtf.org)

*Group Security (GSEC)* [6] working groups to deal with them. This would be another possible area for future work.

**Study of Suitable RAI for Different Applications**

*RAI* is the time interval that ERA adjusts its target rate. It indicates the responsiveness of the protocol. Our experiments in Chapter 6 have used *RAI* of one second just for simulation purposes. In a real implementation, *RAI* would be set according to the nature of applications running on top of ERA. Some elastic applications (such as Web, E-mail, FTP) may prefer a short *RAI* to be very responsive to the network conditions, while some multimedia applications (such as real-time audio/video) may prefer a longer *RAI* to maintain smoothness of reception quality. Future work can be done to discover the optimal *RAI* value of different kinds of multicast applications.

### 7.3.3 Emulating, Prototyping and Measurement on a Test-bed

All the performance studies done in this research have relied only on the network simulation technique. While the network simulation is accepted by the research community as a very useful and significant performance evaluation tool giving sound insights of a system, it should not be misguidedly taken as the real world. Further work would be able to explore our proposed MR-MCC protocol over a real-world experimental test-bed using a wide variety of real Internet applications. Experiments on the test-bed would capture important details that might be missed in our simulation. However, this involves prototyping and building a test-bed, which are expensive. The next step moving towards the measurement on the test-bed may be using network emulators (such as *Dummynet* [144]), which allows a running simulator to interact with operational network nodes. Then, we can move forward testing ERA on an existing global multicast test-bed (MBONE [56]). After all, we note that even

with the availability of a test-bed, its size is limited and not similar to the size of global Internet either. While building the *Internet-Size Test-bed* would not be possible in the near future, the test-bed experiments would give some further insights of ERA.

### 7.3.4 Available Bandwidth Estimation Techniques

ERA uses the available bandwidth estimation technique of Paxson (*Receiver-sided Packet-bunches Probe*), which is one of several techniques to estimate available bandwidth. It is still an open-research issue to get an accurate estimation in practice. Several studies (both in network simulation and in real test-bed), such as [17], [37], [54], [55], [72], [90], [70], [114] and [154], have so far tried to tackle this problem. Nevertheless, there is still much work to be done. So, this would be another path for future work.

### 7.3.5 Fairness Issues between Multicast and Unicast

IETF have forced TCP-friendliness as a fairness paradigm for any new protocols to be deployed over the Internet to protect new types of traffic from starving TCP, the current majority traffic. Hence, ERA has been designed by using this as a goal. However, TCP is a unicast protocol while ERA is a multicast protocol. A session of ERA may serve several receivers, while a TCP session serves only one. Hence, it may not be *so-called fair* to treat a multicast session as a single session deserving no more bandwidth than a single unicast session. To give incentive in deploying multicast instead of *multiple unicast*, we may give the multicast session more bandwidth than TCP (unicast) session. Without this incentive, the overhead of multicast (such as IGMP, more complex security issues and congestion control) would not be attractive. So far, only a little work has been initially done on defining a better fairness paradigm

between multicast and unicast (such as [96] and [162]). Much work is still needed to meet this issue. Defining and evaluating a new *Multicast vs. Unicast Fairness Paradigm* is an interesting future research avenue.

### 7.3.6 Further Study on MR-MCC protocols

While we believe that our simulation studies of MR-MCC protocols (both on the earlier proposals in Chapter 4, and on our new proposed one in Chapter 6) have given a lot of insights, they are still far away from a complete study of protocol behaviours. Our choices of simulation scenarios have been useful to test several aspects of MR-MCC schemes. However, there are other aspects (e.g., analyses of MR-MCC overhead, providing MR-MCC in QoS networks, and analyses of suitable layering schemes and layer granularity with different kinds of applications, etc.) that can be explored further. Performance testing of MR-MCC protocols is still an open-research area where much additional work is required.

### 7.3.7 FEC Error Control and Receiver-driven Layered Congestion Control in Unicast

An interesting idea is to apply the concepts of FEC error control and receiver-driven layered congestion control in unicast as well. With the current progress of FEC technologies, high reliability with a low overhead can be guaranteed. For example, *DF* [32] technology is claimed to be able to provide 99.99% reliability with only 5% of overhead. Hence, FEC could be a competing choice (apart from ARQ) for error control technique in unicast. By using FEC, instead of sending original packets, we can send encoded data packets, which are interchangeable. Which packets received or what order they have been received will not affect the delivery of data as long as the

adequate packets have been received to build the original data. If any encoded data packets are lost, any additionally received packets can replace them. So, this would ease the error control mechanisms in unicast.

Furthermore, together with the receiver-driven congestion control and layered coding techniques, a server can also serve its users on their demands. Multiple servers may also be used to generate encoded packets from a single file. Then, users can tune into more than one server for faster reception. It would be interesting to study how FEC error control and layered receiver-driven congestion control would be applied in unicast communication, and what the pros and cons in comparison with the current error and congestion control technique are.

# Appendix A: Validation of FLID-DL module

In Chapter 4, we simulated two MR-MCC protocols (PLM and FLID-DL) in comparison. PLM test-suite is provided by its authors. We therefore use the test-suite to validate PLM module as mentioned in Chapter 3. However, there is no test-suite provided for FLID-DL. To validate FLID-DL module, we have simulated it within our *test-suite simulation set*. This appendix presents an example of our validation test for FLID-DL.

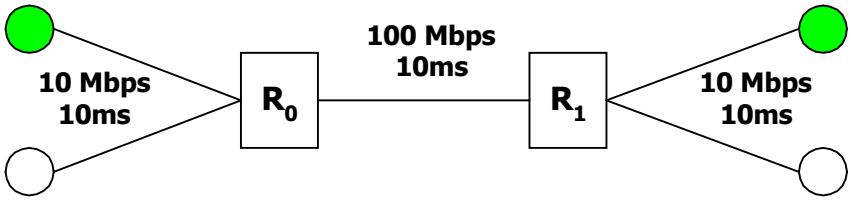



**Figure A-1: Simulation topology of FLID-DL validation**

From Figure A-1, we deploy a dumbbell topology, using a single FLID-DL session sharing with a TCP session. Each exterior link's bandwidth is 10 Mbps, while the middle link is 100 Mbps to ensure no congestion. The delay of each exterior link and the bottleneck link is set to 10 milliseconds. We start both FLID-DL and TCP session randomly during the first 10 seconds. The simulation is run for 100 seconds. TCP and FLID-DL parameters are set to be the same as simulation parameters, mentioned in Chapter 4, Section 4.8.3. Routers in this scenario are drop-tail FIFO with the buffer size of twice bandwidth-delay product (as mentioned in Chapter 3). The packet size is set to 512 bytes.

Theoretically, we expected 12207.03 packets received during the last 50 seconds for TCP. This is calculated from:

```
Maximum Capacity = 10 Mbps
50 seconds of transmission would gain = 50 * 10 = 500 Mb
                                      = 6250000 bytes
Packet Size = 512 bytes
So, we would gain  = 12207.03 packets from the transmission.
```

From the experiment, we get 12207 received packets for TCP during the last 50 seconds. So, it is validated.

For FLID-DL, with *Rate Multiplier (C)* of 1.3, *Base Layer Rate* ($R_o$) of 12 Kbps. We vary the number of layers from 23 to 26. The number of received packets for the last 50 seconds of transmission can be calculated as $\frac{50 * C^i * R_0}{8 * 512}$. From Table A-1, our tests show the results validated within 2% error.

| nLayers | Calculated | Experiment | Error (%) |
|---|---|---|---|
| 26 | 103365.4 | 101484 | 1.82% |
| 25 | 75911.83 | 78049 | 1.84% |
| 24 | 62630.86 | 60398 | 1.25% |
| 23 | 47048.42 | 46832 | 0.46% |

**Table A-1: FLID-DL validation**

# Appendix B: Validation of ERA module: Packet-bunch Probe

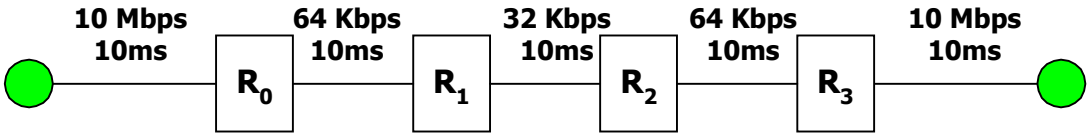



**Figure B-1: Simulation topology of ERA validation**

In order to validate our Packet-bunch Probe implementation, the topology, depicted in Figure B-1, is used. The source sends data in pairs to sink via four routers ($R_0$-$R_3$). The bottleneck link is *link* $R_1$-$R_2$ with capacity of 32 Kbps. The routers are drop-tail FIFO, with the buffer size of twice delay-bandwidth product.

The Packet-bunch Probe in our ERA modules can consistently report $R'_{pp}$ = 32 Kbps. We then shift the bottleneck from the mid-point to link ($R_0$-$R_1$), and link ($R_2$-$R_3$). Our packet-bunch probe implementation still report within an accuracy of 2% in all tests.

# Appendix C: Run-time Trace

---

To ensure that our ns-2 implementation of ERA performs as we designed, we have checked *Run-time Trace*, for every simulation that we have done on ERA. The followings are examples of *Run-time Trace* (used in 6.5.3 Experiment III).

```
line 1: GLVAR inbp = 1e6
line 2: GLVAR btnbp = 400e3
line 3: GLVAR scenario = 4
line 4: GLVAR packetSize = 512
line 5: GLVAR runtime = 200
line 6: GLVAR run_nam = 0
line 7: GLVAR nukDebug = 1
line 8: GLVAR MrtMode = PIM-DM
line 9: GLVAR PP_burst_length = 2
line 10:GLVAR PP_estimation_length = 3
line 11:GLVAR Queue_sched_ = DropTail
line 12:Bottleneck Link : 20ms 400e3
line 13: InLink node 2 : 5ms 1e6
line 14: InLink node 3 : 5ms 1e6
line 15: OutLink node 4 : 5ms 1e6
line 16: OutLink node 5 : 5ms 1e6
line 17: ERA sender on node 2 placed at 0
line 18: ERA receiver on node 4 placed at 1
line 19: ERA Trace 1 for node 4
line 20: flow ID of TCP Reno 1 = 2
line 21: Attach TCP Reno 1 to node 3
line 22: Attach TCP sink 1 to node 5
line 23: Connect TCP Reno 1 and TCP Sink 1
line 24: Attach FTP 1 on TCP reno 1
line 25: FTP 1 start at 20
```

| **Line** | **Comments** |
|---|---|
| 1-11 | Simulation configurations/parameters |
| 7 | Turn run-time trace on |
| 12-16 | Exterior links and bottleneck link configuration |
| 17-25 | Place sources and sinks, ERA at 1 seconds and FTP at 20 seconds |

```
line 26: every RAI, rate adaptation process:
         Time Estimate=2.9029036896856022 Now=2.9545508103419005
line 27: Rpp is min of R'pp:
line 28: List of R'pp = 399999.99999999895 399999.99999999895
         399999.99999999895 399999.99999999895 399999.99999999895
         399999.99999999895 399999.99999999895 399999.99999999895
         399999.99999999895 399999.99999999895 399999.99999999895
         399999.99999999895 399999.99999999895 399999.99999999895
         399999.99999999995 399999.99999999895 399999.99999999895
line 29: Rpp = 399999.99999999895
line 30: hh_npkts=10, hh_nloss=1, npkts=34, nloss=0
line 31: Calculate Rtcp
            PLR=0.0, Rtcp: n/a
line 32: choose_layer(): Rtarget: 400000
line 33: choose_layer(): The Target Layer is 19, R19 = 400000.0
line 34: ERA/ns: 2.95455 node 4 layer 7 ADD-LAYER
line 35: ERA/ns: 2.95455 node 4 layer 8 ADD-LAYER
line 36: ERA/ns: 2.95455 node 4 layer 9 ADD-LAYER
line 37: ERA/ns: 2.95455 node 4 layer 10 ADD-LAYER
line 38: ERA/ns: 2.95455 node 4 layer 11 ADD-LAYER
line 39: ERA/ns: 2.95455 node 4 layer 12 ADD-LAYER
line 40: ERA/ns: 2.95455 node 4 layer 13 ADD-LAYER
line 41: ERA/ns: 2.95455 node 4 layer 14 ADD-LAYER
line 42: ERA/ns: 2.95455 node 4 layer 15 ADD-LAYER
line 43: ERA/ns: 2.95455 node 4 layer 16 ADD-LAYER
line 44: ERA/ns: 2.95455 node 4 layer 17 ADD-LAYER
line 45: ERA/ns: 2.95455 node 4 layer 18 ADD-LAYER
```

| **Line** | **Comments** |
|---|---|
| 26 | At every *RAI*, rate adaptation algorithms start. |
| 27-29 | Calculate $R_{pp}$ from Min of $R'_{pp}$ |
| 30-31 | Calculate $R_{tcp.}$ Yet PLR = 0. So, $R_{tcp}$ is not applicant for target rate. |
| 32-33 | Choose the number of layers to be subscribed |
| 34-45 | Subscribe to the optimal number of layers |

```
line 46: every RAI, rate adaptation process:
         Time Estimate=22.141840432665823 Now=22.153485531323518
line 47: Rpp is min of R'pp:
line 48: List of R'pp = 400000.0000000163 400000.0000000163
         400000.0000000163 400000.0000000163 400000.0000000163
         400000.0000000163 400000.0000000163 400000.0000000163
         400000.0000000163 400000.0000000163 400000.0000000163
         400000.0000000163 400000.0000000163 200000.00000000815
line 49: Rpp = 200000.00000000815
line 50: hh_npkts=863, hh_nloss=2, npkts=828, nloss=4
line 51: Calculate Rtcp
         PLR=0.004807692307692308
         Rtcp = 1386939.5073044538
line 52: choose_layer(): Rtarget: 200000
line 53: choose_layer(): The Target Layer is 9 R9 = 200000.0
line 54: ERA/ns: 22.93768 node 4 layer 18 DRP-LAYER 18
line 55: ERA/ns: 22.93768 node 4 layer 17 DRP-LAYER 17
line 56: ERA/ns: 22.93768 node 4 layer 16 DRP-LAYER 16
line 57: ERA/ns: 22.93768 node 4 layer 15 DRP-LAYER 15
line 58: ERA/ns: 22.93768 node 4 layer 14 DRP-LAYER 14
line 59: ERA/ns: 22.93768 node 4 layer 13 DRP-LAYER 13
line 60: ERA/ns: 22.93768 node 4 layer 12 DRP-LAYER 12
line 61: ERA/ns: 22.93768 node 4 layer 11 DRP-LAYER 11
line 62: ERA/ns: 22.93768 node 4 layer 10 DRP-LAYER 10
```

| **Line** | **Comments** |
|---|---|
| 46 | At every RAI, rate adaptation algorithms start. |
| 47-49 | Calculate $R_{pp}$ from Min of $R'_{pp}$ |
| 50-51 | Calculate $R_{tcp}$ |
| 52-53 | Calculate $R_{target}$, and choose the number of layers to be subscribed |
| 54-62 | Subscribe to the optimal number of layers |

# References


[1] "Actelis MetaLOOP Technology", http://www.actelis.com/solutions/technology, Last accessed: August 2003.

[2] "Internet Domain Survey", Internet Software Consortium, http://www.isc.org/ds, Last accessed: August 2003.

[3] "OPNET Modeler", OPNET Technologies, http://www.mil3.com, Last accessed August 2003.

[4] "Internet Telephone Consortium (ITC)", http://itel.mit.edu, Last accessed: September 2003.

[5] "IETF Multicast Security (MSEC) Charter", http://www.ietf.org/html.charters/msec-charter.html, Last accessed: September 2003.

[6] "IRTF Group Security Charter", http://www.irtf.org/charters/gsec.html, Last accessed: September 2003.

[7] "IETF IP Security Protocol (IPSEC) Charter", IP Security Protocol, Last accessed: September 2003.

[8] "Real Network Homepage", http://www.realnetworks.com, Last accessed: September 2003.

[9] "QuickTime Homepage", http://www.apple.com/quicktime, Last accessed: September 2003.

[10] "Windows Media Player Homepage", http://www.microsoft.com/windows/windowsmedia, Last accessed: September 2003.

[11] "VocalTec Homepage", http://www.vocaltec.com, Last accessed: September 2003.

[12] "Whatis Target Search", http://whatis.com, Last accessed: September 2003.

[13] "RSA Security Homepage", http://www.rsasecurity.com, Last accessed: September 2003.

[14] "Multicast Deployment Made Easy", Cisco, Design Implementation Guide IP Multicast Planning and Deployment Guide, 1998, http://www.cisco.com, Last accessed: May 2001.

[15] "Toasty: a web enabled weather forecasting toaster", http://www.moonfarmer.org/archives/2002_02_07.php Last Accessed: April 2001.

[16] A. Adams, J. Nicholas, and W. Siadak, "Protocol Independent Multicast - Dense Mode PIM-DM): Protocol Specification (Revised)", IETF, Internet Draft (Work in progress, expired August 2003) February 2003.

[17] B. Ahlgren, M. Bjorkman, and B. Melander, "Network Probing Using Packet Trains", Unpublished report, March 1999, http://www.sics.se/~bengta/recent-work.html, Last accessed: August 2003.

[18] S. Ahuja, "COMNETIII: A Network Simulation Laboratory Environment for a Course in Communication Networks", In Proceedings of Frontier in Education (FIE) 1998, Tempe, Arizona, USA, November 1998.

[19] M. Allman, "TCP option deployment", http://roland.grc.nasa.gov/~mallman/tcp-opt-deployment, Last accessed: August 2003.

[20] M. Allman, "A Web Server's View of the Transport Layer", Computer Communication Review, 30(5):133-142, June 2000.

[21] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control", IETF, RFC 2581, April 1999.

[22] M. H. Ammar and G. Rouskas, "On the performance of protocols for collecting responses over a multiple-access channel", IEEE Transactions on Communications, 43(2):410-420, February-April 1995.

[23] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, M. Handley, A. Helmy, J. Heideman, P. Huang, S. Kumar, S. McCanne, R. Rejaie, P. Sharma, K. Varadhan, Y. Xu, H. Yu, and D. Zappala, "Improving Simulation for Network Research", University of Southern California, Technical report 99-702b, September 1999.

[24] D. Bansal, H. Balakrishman, S. Floyd, and S. Shenker, "Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms", In Proceedings of ACM SIGCOMM, San Diego, CA, USA, August 2001.

[25] M. Baumannn, "YATS User's and Programmer's Manual for version 0.3", Dresden University of Technology, September 1997, http://www.ifn.et.tu-dresden.de/TK/yats/yats.html, Last Accessed: September 2003.

[26] P. Bhagwat, P. Misra, and S. Tripathi, "Effect of Topology on Performance of Reliable Multicast Communication", In Proceedings of IEEE INFOCOM 1994, pp. 602-609, Toronto, Canada, June 1994.

[27] S. Bhattacharyya, "An Overview of Source-Specific Multicast (SSM)", IETF, RFC 3569, July 2003.

[28] J. Boyce and R. Galianello, "Packet Loss Effects on MPEG Video Sent over the Internet", In Proceedings of ACM Multimedia, pp. 189-190, Bristol, UK, September 1998.

[29] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", IETF, RFC 2309, April 1998.

[30] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heideman, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in Network Simulation", IEEE Computer Journal, 33(5):59-67, May 2000.

[31] J. W. Byers, M. Luby, and M. Mitzenmacher, "Fine-grained Layered Multicast", In Proceedings of IEEE INFOCOM, pp. 275-283, Anchorage, Alaska, USA, April 2001.

[32] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data", In Proceedings of ACM SIGCOMM, Vancouver, Canada, September 1998.

[33] J. W. Byers, G. Horn, M. Handley, M. Luby, W. Shaver, and L. Vicisano, "More Thoughts on Reference Simulations for Reliable Multicast Congestion Control Schemes", Notes from a meeting at Digital Fountain, August 8, 2000.

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

[35] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, "Internet Group Management Protocol, Version 3", IETF, RFC 3376, October 2002.

[36] K. L. Calvert, M. B. Doar, and E. W. Zegura, "Modeling Internet Topology", IEEE Communications Magazine, 35(6):160-163, June 1997.

[37] R. Carlson, "ANL Web100 based Network Diagnostic Tool (NDT)", http://miranda.ctd.anl.gov:7123, March 20, 2003, Last accessed: September 2003.

[38] S. Casner, "First IETF Internet Audiocast", Computer Communications, 22(3):92-97, July 1992.

[39] S. R. Chandran and S. Lin, "Selective-repeat-ARQ Schemes for Broadcast Links", IEEE Transactions on Communications, 40(1):12-19, January 1992.

[40] J. Chung and M. Claypool, "NS by Examples", http://nile.wpi.edu/NS, Last accessed: June 2001.

[41] K. Claffy, "WAN packet size distribution", http://www.nlanr.net/NA/Learn/packetsizes.html, Last updated: June 1997, Last accessed: September 2003.

[42] K. Claffy and G. J. Miller, "The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone", In Proceedings of INET, July 1998.

[43] K. Claffy, H. W. Braun, and G. Polyzos, "Measurement considerations for assessing unidirectional latencies", Journal of Internetworking, 4(3):14-19 September 1993.

[44] K. Claffy, G. Polyzosand, and H. W. Braun, "Traffic Characteristics of the T1 NSFNET Backbone", In Proceedings of IEEE INFOCOM, San Francisco, USA, March 1993.

[45] J. Crowcroft, "TCP-friendliness Considered Unfriendly", In Proceedings of Multi-Service Networks (MSN), Abingdon, Oxfordshire, UK, July 2001, http://www.acu.rl.ac.uk/msn2001, Last Accessed: September 2001.

[46] A. Day, "How to write & publish a scientific paper", Third Edition, Cambridge University Press, 1989

[47] S. Deering, "Host Extensions for IP Multicasting", IETF, RFC 1112, August 1989.

[48] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", IETF, RFC 2460, December 1998.

[49] D. Delucia and K. Obraczka, "Multicast Feedback Suppression using Representatives", In Proceedings of IEEE INFOCOM, pp. 463-470, April 1997.

[50] D. Delucia and K. Obraczka, "Congestion Control Performance of a Reliable Multicast Protocol", In Proceedings of Network Protocols, pp. 168-176, October 1998.

[51] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm", In Proceedings of ACM SIGCOMM, pp. 1-13, Austin, Texas, USA, September 1989.

[52] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture", IEEE Network Magazine Special Issue on Multicasting, 14(1):78-88, January/February 2000.

[53] M. B. Doar, "A Better Model for Generating Test Networks", In Proceedings of IEEE Global Internet, pp. 86-93, London, UK, November 1996.

[54] C. Dovrolis, P. Ramanathan, and D. Moore, "What do Packet Dispersion Technique Measure?", In Proceedings of IEEE INFOCOM, Anchorage, Alaska, USA, April 2001, http://www.cc.gatech.edu/~dovrolis/Papers/infocom01.ps, Last Accessed: September 2003

[55] C. Dovrolis, M. Jain, and R. S. Prasad, "BW-meter: Measurement tools for the capacity and load of Internet paths", http://www.pathrate.org, Last accessed: September 2003.

[56] H. Eriksson, "MBONE: The Multicast Backbone", Communications of the ACM, 37(8):54-60, August 1994.

[57] A. Erramilli and R. P. Singh, "A Reliable and Efficient Multicast Protocol for Broadband Broadcast Networks", In Proceedings of ACM SIGCOMM 1988, pp. 343-352, August 1988.

[58] W. Fenner, "Internet Group Management Protocol version 2", IETF, RFC 2236, January 1997.

[59] S. Floyd, "Congestion Control Principles", IETF, RFC 2914, September 2000.

[60] S. Floyd, "Simulation is crucial", Appeared as a side bar in IEEE Spectrum January 2001.

[61] S. Floyd and V. Jacobson, "Random Early Detection gateways for congestion avoidance", IEEE/ACM Transactions on Networking, 1(4):397-401, August 1993.

[62] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", IEEE/ACM Transactions on Networking, 7(4):458-472, August 1999.

[63] S. Floyd and V. Paxson, "Difficulties in Simulating the Internet", IEEE/ACM Transactions on Networking, 9(4):392-403, August 2001.

[64] R. Gass, "Packet Trace Analysis by IP Monitor Project", http://ipmon.sprint.com/packstat/packetoverview.php, Last accessed: July 2003.

[65] GloMoSim, "Global Mobile Simulator", on-line software: http://pcl.cs.ucla.edu/projects/glomosim, Last accessed: July 2003.

[66] R. Gopalakrishman, J. Griffoen, G. Hjalmtysson, C. J. Sreenan, and S. Wen, "A simple loss differentiation approach to layered multicast", In Proceedings of IEEE INFOCOM, pp. 461-469, March 2000.

[67] S. Gorinsky, S. Jain, and H. Vin, "Robustness to Inflated Subscription in Multicast Congestion Control", In Proceedings of ACM SIGCOMM, pp. 87-98, Karlsruhe, Germany, August 2003.

[68] R. Goyal, R. Jain, S. Fahmy, and B. Vandalore, "Buffer Management for TCP over the ATM GFR Service", ATM Forum, ATM98-0405, July 1998, http://www.netlab.ohio-state.edu/~jain/atmf/a98-0405.htm, Last Accessed: September 2001.

[69] K. Harfoush, A. Bestavros, and J. Byers, "PeriScope: An Active Internet Probing and Measurement API", Computer Science Department, Boston University, Technical Report 2002-005, May 2002, http://www.cs.bu.edu/techreports/abstracts/2002-005, Last Accessed: September 2003.

[70] K. Harfoush, A. Bestavros, and J. Byers, "Measuring Bottleneck Bandwidth of Targeted Path Segments", In Proceedings of IEEE INFOCOM, April 2003, http://www.cs.bu.edu/techreports/pdf/2002-005-periscope.pdf, Last Accessed: September 2003.

[71] H. W. Holbrook, S. K. Singhal, and D. R. Cheriton, "Log-based receiver-reliable multicast for distributed interactive simulation", In Proceedings of ACM SIGCOMM 1995, Cambridge, USA, August-September 1995.

[72] N. Hu and P. Steenkiste, "Initial Gap Increasing (IGI) and Packet Transmission Rate (PTR)", http://www.cs.cmu.edu/~hnn/igi, March 10, 2003, Last accessed: September 2003.

[73] G. Huston, "TCP Performance", The Internet Protocol Journal, 3(2):2-24, June 2000, http://www.cisco.com/ipj, Last Accessed: September 2003.

[74] C. Iancu and A. Acharya, "A Comparison of Feedback Based and Fair Queuing Mechanism for Handling Unresponsive Traffic", Unpublished manuscript, 2001,http://citeseer.nj.nec.com/461118.html, Last Accessed: September 2002.

[75] V. Jacobson, "Congestion Avoidance and Control", In Proceedings of Symposium on Communications Architecture and Protocols (SIGCOMM 1988), pp. 314-329, Stanford, CA, USA, August 1988.

[76] V. Jacobson, R. Braden, and D. Borman, "TCP Extensions for High Performance", IETF, RFC 1323, May 1992.

[77] V. Jacobson, S. McCanne, and S. Floyd, "Lightweight Sessions - A new architecture for real-time applications and protocols", Third annual principal investigation meeting, ARPA, Santa Rosa, CA, USA, September 1993.

[78] J. M. Jaffle, "Bottleneck Flow Control", IEEE Transactions on Communications, 7(9):954-962, July 1981.

[79] R. Jain, "Congestion Control in Computer Networks: Issues and Trends", IEEE Network, 4(3):24-30, May 1990.

[80] R. Jain, "The Art of Computer Systems Performance Analysis", John Wiley, 1991.

[81] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System", Digital Equipment Corporation, Hudson, MA 01749, Technical Report DEC-TR-301, 1984.

[82] K. Kang, D. Lee, H. Y. Youn, and K. Chon, "NLM: Network-based Layered Multicast for traffic control of heterogeneous network", Computer Communications, March 2001.

[83] D. Katz, "A Proposed Standard for the Transmission of IP Datagrams over FDDI Networks", IETF, RFC 1103, June 1989.

[84] F. Kelly, A. Maulloo, and D. Tan, "Rate Control in Communication Network: shadow price, proportional fairness and scalability", Journal of the Operational Research Society, 49:237-252, 1998.

[85] S. Keshav, "REAL: a Network Simulator", University of California Berkeley, Technical Report 88-472, 1988.

[86] S. Keshav, "A Control-Theoretic Approach to Flow Control", In Proceedings of ACM SIGCOMM, pp. 3-15, Zurich, Switzerland, September 1991.

[87] S. Keshav, "Congestion Control in Computer Networks", PhD Thesis, EECS Department, UC Berkeley, 1991.

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

[89] J. F. Kurose and K. W. Ross, "Computer Networking - a Top Down Approach Featuring the Internet", Second Edition, Addison Wesley, 2003

[90] K. Lai, "Measuring the Bandwidth of Packet Switched Networks", PhD Thesis, Department of Computer Science, Stanford University, 2002

[91] M. Laubach and J. Halpern, "Classical IP and ARP over ATM", IETF, RFC 2225, April 1998.

[92] A. M. Law and W. D. Kelton, Simulation Modeling and Analysis, Second Edition, McGraw-Hill, 1991.

[93] A. Legout and E. W. Biersack, "Beyond TCP-friendliness: A new paradigm for end-to-end congestion control", Institut Eurocom, Sophia Antipolis, France, Technical report, October 1999.

[94] A. Legout and E. W. Biersack, "Pathological Behaviours for RLM and RLC", In Proceedings of NOSSDAV, pp. 164-172, Chapel Hill, North Corolina, USA, June 2000.

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

[96] A. Legout, J. Nonnenmacher, and E. W. Biersack, "Bandwidth-allocation Policies for Unicast and Multicast Flows", IEEE/ACM Transactions on Networking, 9(4):464-478, August 2001.

[97] B. N. Levine, D. B. Lavo, and J. J. Garcia-Luna-Aceves, "The Case for Reliable Concurrent Multicasting using shared ACK trees", In Proceedings of

ACM International Conference on Multimedia, pp. 365-376, Boston, USA, November 1996.

[98] D. Li and D. R. Cheriton, "Evaluating the utility of FEC with reliable multicast", In Proceedings of Seventh International Conference on Network Protocols (ICNP), pp. 97-105, October-November 1999.

[99] S. Lin and D. J. Costello, "Error Correcting Coding: Fundamentals and Applications", Prentice Hall, 1983.

[100] S. H. Low and D. Lapsley, "Optimization Flow Control", IEEE/ACM Transactions on Networking, 7(6):861-875, December 1999.

[101] M. Luby, "Information Additive Code Generator and Decoder for Communication Systems", U.S. Patent, No. 6373406, April 16, 2002.

[102] M. Luby, V. K. Goyal, and S. Skaria, "Wave and Equation-based Rate Control: A Massively Scalable Receiver Driven Congestion Control Protocol", Computer Communications, 32(4):191-214, October 2002.

[103] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Effective Erasure Correcting Codes", IEEE Transactions on Information Theory, Special Issues: Codes on Graphs and Iterative Algorithms, 47(2):569-584, 2001.

[104] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "Forward Error Correction Building Block", IETF, RFC 3452, December 2002.

[105] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, and J. Crowcroft, "Layered Coding Transport (LCT) Building Block", IETF, RFC 3451, December 2002.

[106] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "The Use of Forward Error Correction in Reliable Multicast", IETF, RFC 3453, December 2002.

[107] J. Mahdavi and S. Floyd, "TCP-friendly Unicast Rate-based Flow Control", Technical note sent to the end2end-interest mailing list, January 1997, http://www.psc.edu/networking/papers/tcpfriendly.html.

[108] G. Malkin, "RIP Version 2 Carrying Additional Information", IETF, RFC 1388, January 1993.

[109] A. Mankin and M. Handley, "Reference Simulations for RM Congestion Control", Technical note in RMRG Meeting, UCL, London July 1998, http://www.east.isi.edu/rm/london-meeting.htm, Last Accessed: September 2003.

[110] A. Mankin, A. Romanow, S. Bradner, and V. Paxson, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols", IETF, RFC 2357, June 1998.

[111] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options", IETF, RFC 2018, October 1996.

[112] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The Macroscopic Behaviour of the TCP Congestion Avoidance Algorithm", Computer Communication Review, 27(3):67-82, July 1997.

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

[114] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks", In Proceedings of IEEE Globecom, November 2000.

[115] J. Mogul, "Observing TCP Dynamics in Real Networks", DEC Western Research Laboratory, California, USA, Research Report 92/2, April 1992.

[116] J. Moy, "OSPF Version 2", IETF, RFC 2328, April 1998.

[117] J. Nagle, "Congestion Control in IP/TCP Internetworks", IETF, RFC 896, January 1984.

[118] K. Najafi, "Modeling, Routing and Architecture in Active Networks", PhD Thesis, Department of Electrical and Computer Engineering, The University of Toronto, Canada, 2001.

[119] J. Nonnenmacher and E. W. Biersack, "Optimal multicast feedback", In Proceedings of IEEE INFOCOM, San Francisco, USA, March-April 1998.

[120] J. Nonnenmacher and E. W. Biersack, "Scalable Feedback for Large Groups", IEEE/ACM Transactions on Networking, 7(3):375-386, June 1999.

[121] ns-2, "Network Simulator -- ns version 2", online software: http://www.isi.edu/nsnam/ns, Last Accessed: September 2003.

[122] J. K. Ousterhout, Tcl and the Tk toolkit: Addison-Wesley, 1994

[123] J. D. Padhye, "Model-based Approach to TCP-friendly Congestion Control", PhD Thesis, University of Massachusetts Amherst, USA, 2000.

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

[125] S. Paul, K. K. Sabnani, and D. M. Kristol, "Multicast Transport Protocols for High-Speed Networks", In Proceedings of IEEE International Conference on Network Protocols (ICNP), pp. 4-14, Boston, MA, USA, October 1994.

[126] S. Paul, K. K. Sabnani, and J. C. Lin, "Reliable Multicast Transport Protocol (RMTP)", IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multipoint Communication, 15(3):407-421, April 1997.

[127] K. Pawlikowski, H. J. Jeong, and J. R. Lee, "On Credibility of Simulation Studies of Telecommunication networks", IEEE Communications Magazine, 40(1):132-139, January 2002.

[128] V. Paxson, "Automated Packet Trace Analysis of TCP Implementation", In Proceedings of ACM SIGCOMM, pp. 167-179, September 1997.

[129] V. Paxson, "End-to-end Internet Packet Dynamics", Computer Communications, 27(4):139-152, October 1997.

[130] V. Paxson, "Measurements and Analysis of End-to-end Internet Dynamics", PhD thesis, Lawrence Berkeley National Laboratory, University of California, Berkeley, California, USA, 1997.

[131] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and Secure Source Authentication for Multicast", In Proceedings of Network and Distributed System Security Symposium (NDSS), pp. 35-46, February 2001.

[132] S. Pingali, D. F. Towsley, and J. F. Kurose, "A comparison of Sender-initiated and Receiver-initiated Reliable Multicast Protocols", IEEE Journal on Selected Areas in Communications (JSAC), 15(3):398-406, April 1997.

[133] J. Postel, "User Datagram Protocol", IETF, RFC 768, August 1980.

[134] J. Postel, "Internet Protocol", IETF, RFC 791, September 1981.

[135] J. Postel, "The TCP Maximum Segment Size and Related Topics", IETF, RFC 879, November 1983.

[136] S. Puangpronpitag, "Somnuk Puangpronpitag PhD Research Log Web Page", http://www.comp.leeds.ac.uk/nuk/Research/Log, Last updated: August 2003, Last accessed: September 2003.

[137] S. Puangpronpitag, "Performance Analysis of TCP Congestion Control over ATM Networks", MSc Thesis, School of Computer Studies, University of Leeds, 1999.

[138] S. Puangpronpitag, M. Kara, and K. Djemame, "A Performance Evaluation of Buffer Management and Scheduling for ATM-GFR using TCP", In Proceedings of IFIP Workshop on Performance Modelling and Evaluation of ATM and IP Networks, pp. 71/1-71/14, Ilkley, UK, July 2000.

[139] S. Puangpronpitag, M. Kara, and K. Djemame, "ATM-GFR Buffer Management and Scheduling Issues: a Performance Study using TCP", In

Proceedings of the UK Performance Engineering Workshop, pp. 191-202, Leeds, UK, July 2001.

[140] S. Puangpronpitag, R. D. Boyle, and K. Djemame, "Performance Evaluation of Layered Multicast Congestion Control Protocols: FLID-DL vs. PLM", In Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 03), Montreal, Canada, July 2003.

[141] I. Rhee, N. Balaguru, and G. Rouskas, "MTCP: Scalable TCP-like Congestion Control for Reliable Multicast", In Proceedings of IEEE INFOCOM, vol. 3, pp. 1265-1273, March 1999.

[142] R. Rivest, "The MD5 Message-Digest Algorithm", IETF, RFC 1321, April 1992.

[143] L. Rizzo, "Luigi Rizzo's Research Web Page", http://info.iet.unipi.it/~luigi/research.html, Last accessed: September 2003.

[144] L. Rizzo, "Dummynet: a Simple Approach to the Evaluation of Network Protocols", Computer Communication Review, 27(1):31-41, January 1997.

[145] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols", Computer Communication Review, 27(2):24-36, April 1997.

[146] L. Rizzo, "Fast Group Management in IGMP", In Proceedings of Hipparch Workshop, pp. 32-41, London, UK, June 1998.

[147] L. Rizzo, "PGMCC: A TCP-friendly Single-rate Multicast Congestion Control Scheme", In Proceedings of SIGCOMM, pp. 17-28, Stockholm, Sweden, August 2000.

[148] R. F. Sari and K. Djemame, "Performance Comparison of Active and Non-active Network-based Multicast Multirate Congestion Control Protocols", In Proceedings of IEEE ICON 2002, pp. 249-254, Singapore, August 2002.

[149] K. Seada and A. Helmy, "Fairness Analysis of Multicast Congestion Control: a Case Study on PGMCC", Computer Science Department, University South California, USA, Technical Report 01743, April 2001.

[150] P. Sharma and L. Wei, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification", IETF, RFC 2362, June 1998.

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

[152] K. Sripanidkulchai, A. Myers, and H. Zhang, "A Third-party Value-Added Network Service Approach to Reliable Multicast", In Proceedings of SIGMETRIC 1999, Atlanta, USA, 1999.

[153] P. Stathopoulos, V. Zoi, D. Loukatos, L. Sarakis, and N. Mitrou, "A Network-Driven Architecture for the Multicast Delivery of Layered Video and A Comparative Study", In Proceedings of IFIP Workshop on Internet Technologies, Applications and Social Impact (WITASI), pp. 140-154, Wroclaw, Poland, October 2002.

[154] J. Sterbenz, C. Patridge, M. Allman, D. Rockwell, and R. Krishman, "Swift Start Proposal", http://www.ir.bbn.com/projects/pace/cap/index.html, Last accessed: September 2003.

[155] A. Tanenbaum, Computer Networks, Fourth Edition, Prentice Hall, 2003

[156] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area Internet Traffic Patterns and Characteristics", IEEE Network, 11(6): 10-23, December 1997.

[157] J. Touch, J. Heidemann, and K. Obraczka, "Analysis of HTTP Performance", USC/Information Science, Institute Report Initial Release V1.2, August 1996.

[158] D. F. Towsley, "An Analysis of a Point-to-Multipoint Channel using a Go-Back-N Error Control Protocol", IEEE Transactions on Communications, vol. 33, pp. 282-285, March 1985.

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

[160] R. Wade, "The Design and Simulation of a Transport Protocol for Interactive Network Applications", PhD Thesis, School of Computing, University of Leeds, Leeds, UK, 2000.

[161] D. Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol", IETF, RFC 1075, November 1988.

[162] H. A. Wang and M. Schwartz, "Achieving Bounded Fairness for Multicast and TCP Traffic in the Internet", In Proceedings of ACM SIGCOMM 1998, pp. 81-92, Vancouver, Canada, September 1998.

[163] C. Weinstein and J. Forgie, "Experience with speech communication in packet networks", IEEE Journal on Selected Areas in Communications (JSAC), vol. 1, No. 6, pp. 963-980, December 1983.

[164] J. Widmer, "Equation-based Congestion Control", MSc Thesis, Department of Mathematics and Computer Science, University of Mannheim, Germany, 2000.

[165] J. Widmer and M. Handley, "Extending Equation-based Congestion Control to Multicast Application", In Proceedings of ACM SIGCOMM, pp. 275-286, San Diego, CA, USA, August 2001.

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

[167] R. Wilder, K. Ramakrishnan, and A. Mankin, "Dynamics of Congestion Control and Avoidance of Two-way traffic in an OSI testbed", Computer Communication Review, 21(2):43-58, September 1991.

[168] L. Yamamoto, "Adaptive Group Communication over Active Networks", PhD Thesis, Faculty of Science, University of Liege, Belgium, 2002.

[169] L. Yamamoto and G. Leduc, "Adaptive Applications over Active Networks: Case Study on Layered Multicast", In Proceedings of European Conference on Universal Multiservice Network, pp. 386-394, Colmar, France, October 2000.

[170] R. Yavatkar, J. Griffoen, and M. Sudan, "A Reliable Dissemination Protocol for Interactive Collaborative Applications", In Proceedings of ACM Multimedia, pp. 330-334, San Francisco, CA, USA, November 1995.

[171] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to Model an Internetwork", In Proceedings of IEEE INFOCOM, pp. 594-602, San Francisco, CA, USA, March 1996.

[172] E. W. Zegura, K. L. Calvert, and M. J. Donahoo, "A Quantitative Comparison of Graph-based Models for Internet Topology", IEEE/ACM Transactions on Networking, 5(6):770-783, December 1997.

[173] L. Zhang, S. Shenker, and D. Clark, "Observations on the Dynamics of a Congestion Control Algorithm: the Effect of Two-way Traffic", In Proceedings of ACM SIGCOMM, pp. 133-147, September 1991.

[174] L. Zhu, N. Ansari, Z. Sahinoglu, A. Vetro, and H. Sun, "Scalable Layered Multicast with Explicit Congestion Notification", In Proceedings of IEEE International Conference Information Technology Coding and Computing, April 2003.