




## Approvals

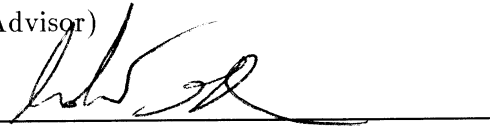
This dissertation is submitted in partial fulfillment of the requirements for the degree of  
Doctor of Philosophy  
Computer Science

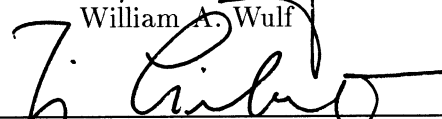
  
Matthew T. Lucas

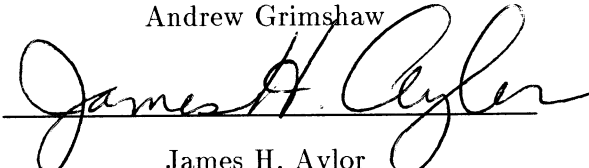
Approved:

  
Alfred C. Weaver (Advisor)


  
William A. Wulf

  
Andrew Grimshaw

  
Jörg Liebeherr (Chair)

  
James H. Aylor  
(Minor Representative)

Accepted by the School of Engineering and Applied Science:

  
Richard W. Miksad (Dean)

May 1998

# **Efficient Data Distribution in Large-Scale Multicast Networks**

A Dissertation

Presented to the Faculty of the School of Engineering and Applied Science  
University of Virginia

In Partial Fulfillment of the Requirements for the Degree of  
Doctor of Philosophy  
Computer Science

by

**Matthew T. Lucas**

May 1998

Copyright © 1998 by Matthew T. Lucas

## Abstract

---

There is an increasing demand for using today's shared computer networks, such as the Internet, for group-based applications. *Multicast* is a developing communication technology designed to provide efficient multi-point message delivery for large-scale groups. Research directed at the data link and network layers has been very successful, and multicast service is now available in most best-effort networks. However, best-effort multicast networks do not offer quality of service guarantees such as bounded transmission delays and error rates. Therefore, group-based applications rely on *multicast transport protocols* for ordering, reliability, group management, and other end-to-end services.

This dissertation presents MESH: a novel, distributed transport protocol designed for large-scale multicast. We show that MESH's error recovery and receiver feedback service (1) achieves high application performance (i.e., low delivery latency and high throughput), (2) efficiently utilizes network and end-system resources, (3) provides a flexible error control model suitable for reliable, unreliable, and other error control paradigms, (4) provides timely state information required by congestion, flow, group management, reliability, and other control protocols, and (5) scales to large receiver sets and wide-area heterogeneous networks. Using the MESH framework, we design and implement a reliable protocol (MESH-R) and a deadline-driven reliable protocol (MESH-M) in a high-fidelity simulation of SURAnet and vBNSnet. We show that the performance and overhead of MESH-R and MESH-M compares favorably to extant transport techniques (namely, centralized, tree-based, unstructured, and FEC-based schemes) for bulk-data distribution and continuous media applications.

*To Tricia, Danielle, Justin, and the little one on the way*

# Contents

---

<b>Abstract</b>	<b>iii</b>
<b>Acknowledgments</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background: Network Support for Multicast . . . . .	3
1.1.1 Data Link Multicast . . . . .	4
1.1.2 Network Layer Multicast . . . . .	4
1.2 Background: Transport Support for Multicast . . . . .	6
1.3 Introduction to Previous Work in Multicast Transport . . . . .	8
1.4 Our Approach: MESH . . . . .	9
1.5 Structure of Dissertation . . . . .	10
<b>2 Multicast Transport Protocols: Previous Work</b>	<b>11</b>
2.1 Multicast Ordering Protocols . . . . .	11
2.2 Reliable Multicast Protocols . . . . .	13
2.3 Real-Time Multicast Protocols . . . . .	14
2.4 Distributed Multicast Transport Protocols . . . . .	17
2.4.1 Distributed Approach at the Network Layer . . . . .	18
2.4.2 Tree-Based Protocols . . . . .	19
2.4.3 Unstructured Protocols . . . . .	22

2.5	The MESH Framework . . . . .	22
2.5.1	MESH Domains . . . . .	23
2.5.2	MESH State Synchronization . . . . .	24
2.5.3	MESH Error Control . . . . .	25
2.6	Summary . . . . .	26
<b>3</b>	<b>Wide-Area Multicast Network Simulation</b>	<b>27</b>
3.1	Introduction to Computer Network Simulation . . . . .	29
3.2	SURAnet and vBNSnet Network Model . . . . .	30
3.3	Campus and Local Area Traffic Model . . . . .	34
3.4	Wide-Area Background Traffic Model . . . . .	39
3.5	WAN Statistical Characterization Experiment Setup . . . . .	40
3.5.1	Experiment Setup . . . . .	41
3.5.2	Data Considered . . . . .	42
3.6	Modeling the Packet Size . . . . .	43
3.7	Aggregate Wide-Area Network Traffic Model, $\mathcal{M}$ . . . . .	46
3.7.1	Time-Dependent Statistical Properties . . . . .	46
3.7.2	Generating Self-Similar Traffic . . . . .	48
3.7.3	Modelling the Arrival Density . . . . .	48
3.7.4	$\mathcal{M}$ Evaluation . . . . .	50
3.8	Partitioning Model $\mathcal{P}$ and Short-term Arrival Model $\mathcal{S}$ . . . . .	52
3.8.1	Partitioning Model $\mathcal{P}$ . . . . .	52
3.8.2	$\mathcal{S}$ : Short-term Packet Arrival Model . . . . .	53
3.9	Application of the $(\mathcal{M}, \mathcal{P}, \mathcal{S})$ Model . . . . .	54
3.10	SURAnet and vBNSnet Simulation Parameters . . . . .	58
3.10.1	SURAnet and vBNSnet $(\mathcal{M}, \mathcal{P}, \mathcal{S})$ Model Parameters . . . . .	59
3.10.2	SURAnet and vBNSnet Delay and Drop Rates . . . . .	63
3.11	Summary . . . . .	68

<b>4</b>	<b>MESH-R: Large-Scale, Reliable Multicast Transport</b>	<b>70</b>
4.1	Motivation . . . . .	72
4.2	Related Work . . . . .	74
4.2.1	Analysis of Control Structure Classes . . . . .	75
4.3	MESH-R Protocol Design and Motivation . . . . .	78
4.3.1	MESH Framework Overview and Motivation . . . . .	78
4.3.2	MESH-R Group State Synchronization Protocol . . . . .	79
4.3.3	MESH-R Error Control Protocol . . . . .	83
4.4	MESH-R Performance Analysis . . . . .	88
4.4.1	Experiment Setup . . . . .	89
4.4.2	Centralized, Tree-based, and Unstructured Protocol Descriptions . . . . .	90
4.4.3	Experimental Results . . . . .	97
4.5	Summary and Future Work . . . . .	110
4.5.1	Future Work on Reducing MESH-R's State Overhead . . . . .	111
4.5.2	Future Work on MESH-R's State Synchronization Protocol . . . . .	112
4.5.3	Future Work on MESH-R's Error Control . . . . .	113
4.5.4	Future Work on Multi-Source Applications . . . . .	114
4.5.5	Future Work on Scalability . . . . .	115
<b>5</b>	<b>MESH-M – Large-scale Transport for Multimedia Applications</b>	<b>116</b>
5.1	Introduction . . . . .	116
5.2	Distributed Retransmission Error Recovery . . . . .	118
5.2.1	The MESH Protocol Framework . . . . .	119
5.2.2	Error Recovery in the Local Area . . . . .	120
5.2.3	Error Recovery in the Wide Area . . . . .	120
5.2.4	Advertisement Protocol . . . . .	121
5.2.5	Retransmission Protocol . . . . .	123
5.3	MESH-M Performance Evaluation . . . . .	123

5.3.1	Simulation Design . . . . .	124
5.3.2	MESH-M Retransmission Behavior . . . . .	126
5.3.3	Comparison of MESH-M and Source-Based FEC . . . . .	128
5.4	Conclusions and Future Work . . . . .	133
<b>6</b>	<b>Conclusions and Future Work</b>	<b>134</b>
6.1	Conclusions . . . . .	134
6.1.1	Large-Scale Reliable Transport: MESH-R . . . . .	135
6.1.2	Large-Scale Deadline-Driven Transport: MESH-M . . . . .	136
6.2	Future Work . . . . .	137
<b>A</b>	<b>Self-Similar Traffic</b>	<b>139</b>
A.1	Generating Self-Similar Traffic Using the FFT Method . . . . .	139
<b>B</b>	<b>Simulated Network Delay and Error Rates</b>	<b>141</b>
B.1	Simulated Round-trip Delays and Drop Rates in SURAnet . . . . .	141
B.2	Simulated Round-trip Delays and Drop Rates in vBNSnet . . . . .	143



## Acknowledgments

---

It is an honor to have Alfred Weaver as my advisor, colleague, and friend throughout the development of this work. Alf recognized the importance and challenge of wide-area multicast, and gave me the freedom to explore my own solutions. I sincerely appreciate his support, insight, direction, and wisdom.

Special thanks to Bert Dempsey and Dallas Wrege. Bert is an exceptional researcher and friend. I thoroughly enjoyed my time writing papers and exploring networking ideas with him. Likewise, special thanks to Dallas (a.k.a. Dr. Figure) for his insights on traffic modeling, and expertise in TeX. It has been a pleasure working with him.

Thanks goes to Jörg Liebeherr, William Wulf, Andrew Grimshaw, and James Aylor who served on my dissertation committee and provided valuable feedback. It is an honor to have this prestigious faculty review and accept my work.

I am also grateful to Paco Hope for helping me with the compute servers and Sudhir Srinivasan for teaching me SES Workbench optimizations. I wish all of you (Alf, Bert, Dallas, Jorg, Bill, Andrew, Jim, Paco, and Sudhir) the very best of luck in your endeavors.

My parents, Diane and Jerome Lucas, deserve special recognition for stressing the importance of education. Their academic accomplishments, as well as their support, have provided me continuous inspiration.

Finally, thanks goes to my wife Tricia for supporting me in my academic pursuits. Tricia is a wonderful friend, wife, mother, and partner to tolerate and encourage the long hours required to complete this work.

## List of Figures

---

2.1	Subgrouping Example of SURAnet . . . . .	20
2.2	Internet domain hierarchy . . . . .	23
3.1	SURAnet infrastructure and topology. . . . .	31
3.2	vBNS infrastructure and topology. . . . .	32
3.3	Campus network model. . . . .	32
3.4	Roundtrip times over a 2 hop path within UVAnet. . . . .	35
3.5	Roundtrip times over a 3 hop path within UVAnet. . . . .	35
3.6	Campus background traffic approach. . . . .	36
3.7	Packet drop series at Olsson Hall router Ethernet interfaces 2, 8, and 12. . . . .	37
3.8	$(\mathcal{M}, \mathcal{P}, \mathcal{S})$ traffic model. . . . .	40
3.9	Experiment Setup . . . . .	42
3.10	Packets per 100 seconds for 9 day packet trace. . . . .	43
3.11	Packets per 10 second interval for first Tuesday packet trace. . . . .	44
3.12	Probability density function of packet sizes. . . . .	45
3.13	Autocorrelation function for packet sizes. . . . .	46
3.14	Autocorrelation function for the 2AM, 3PM and 9PM traces. . . . .	47
3.15	Q-Q plots of 2AM, 9PM and 3PM empirical traces versus fitted log-normal distributions. . . . .	49
3.16	Arrival distribution of empirical traces (solid lines) and synthetic (dashed lines) for low, medium, and high network utilizations. . . . .	51

3.17	Log variance plot of synthetic (green lines) versus empirical (red lines) for 2AM, 9PM and 3PM UVAnet traces. . . . .	51
3.18	Packet-train time series for (a) 2AM, (b) 9PM and (c) 3PM campus substreams with addresses 192.0.0.0 - 199.255.255.255. . . . .	55
3.19	Number of packets in a train plotted against the mean of the campus substream (in packets per 1ms). . . . .	56
3.20	Probability density function for empirical campus streams (solid lines) and model $P$ (dashed lines) for 3PM trace. . . . .	58
3.21	Simple partitioning approach for 3PM trace using empirical probability. . .	59
3.22	Log-variance plot of 3PM component substreams. Note the correlation structure of the synthetic (b) match the empirical (a). . . . .	60
3.23	SURAnet round-trip delays for $\overline{M} = 10 - 70$ background packets per 100 ms. . . . .	65
3.24	SURAnet round-trip drop rates for $\overline{M} = 10 - 70$ background packets per 100 ms. . . . .	66
3.25	vBNSnet round-trip delays for $\overline{M} = 700 - 2700$ background packets per 100 ms. . . . .	67
3.26	vBNSnet round-trip drop rates for $\overline{M} = 700 - 2700$ background packets per 100 ms. . . . .	68
4.1	SURAnet network with multicast source located at CTV, and group members located at each campus network. . . . .	76
4.2	MESH's state synchronization approach. . . . .	80
4.3	MESH-R message format. . . . .	81
4.4	MESH-R error control protocol state diagram. . . . .	84
4.5	MESH-R procedure for selecting a retransmission server. . . . .	86
4.6	Centralized error/control flow example. . . . .	91
4.7	Tree-based error/control flow example. . . . .	93
4.8	SRM adaptive timer algorithm. . . . .	96

**List of Figures** xii

4.9	File transfer times for 1 MB file in SURAnet. . . . .	97
4.10	File transfer times for 10 MB file in vBNSnet. . . . .	98
4.11	Protocol overhead in SURAnet. . . . .	100
4.12	Protocol overhead in vBNSnet. . . . .	100
4.13	Protocol efficiency in SURAnet. . . . .	101
4.14	Protocol efficiency in vBNSnet. . . . .	102
4.15	MESH error recovery example in campus domain. . . . .	106
4.16	Retransmission distribution in vBNSnet between adjacent campus networks. . . . .	108
4.17	Retransmission distribution in SURAnet between adjacent campus networks. . . . .	109
4.18	Example MESH state synchronization. . . . .	112
5.1	S-ARQ retransmission model. . . . .	119
5.2	Advertisement packet format. . . . .	122
5.3	Retransmission procedure for selecting remote AR. . . . .	124
5.4	Wide-area network topology used in simulation experiments. . . . .	125
5.5	Distribution of retransmission requests. . . . .	127
5.6	Retransmission protocol overhead distribution and drop rates per link. . . . .	127
5.7	Application performance under FEC and MESH-M error control. . . . .	129
5.8	Network cost for FEC overcoding. . . . .	130
5.9	Application performance for FEC(4:2), FEC(5:1), and MESH-M with $\mu = 1$ . . . . .	131
5.10	Network cost for FEC(4:2), FEC(5:1), and MESH-M with $\mu = 1$ . . . . .	131
5.11	Application performance for FEC(4:2), FEC(5:1), and MESH-M with $\mu = 3$ . . . . .	132
5.12	Network cost for FEC(4:2), FEC(5:1), and MESH-M with $\mu = 3$ . . . . .	132

## List of Tables

---

3.1	Olsson router utilization and drop statistics. . . . .	38
3.2	Parameters used to model UVAnet arrivals. . . . .	50
3.3	Network filter mask and percent of traffic for 2AM, 9PM and 3PM traces. . . . .	57
3.4	Network address ranges used to define SURAnet substreams. . . . .	62
3.5	Network address ranges used to define vBNSnet substreams. . . . .	63
4.1	vBNSnet AR selection distribution. . . . .	107
B.1	Suranet round-trip network delay (in <i>ms</i> ) for $\overline{M} = 10$ packets per 100 <i>ms</i> . . . . .	141
B.2	Suranet round-trip network delay (in <i>ms</i> ) for $\overline{M} = 70$ packets per 100 <i>ms</i> . . . . .	142
B.3	Suranet round-trip network drop percent for $\overline{M} = 10$ packets per 100 <i>ms</i> . . . . .	142
B.4	Suranet round-trip network drop percent for $\overline{M} = 70$ packets per 100 <i>ms</i> . . . . .	143
B.5	vBNSnet round-trip delays (in <i>ms</i> ) for $\overline{M} = 700$ packets per 100 <i>ms</i> . . . . .	143
B.6	vBNSnet round-trip delays (in <i>ms</i> ) for $\overline{M} = 2700$ packets per 100 <i>ms</i> . . . . .	144
B.7	vBNSnet round-trip drop percent for $\overline{M} = 700$ packets per 100 <i>ms</i> . . . . .	144
B.8	vBNSnet round-trip drop percent for $\overline{M} = 2700$ packets per 100 <i>ms</i> . . . . .	144

## Introduction

---

Recently, there has been a massive investment in computer network infrastructure to support the growing communication demands of networked computer systems. Although computer networks were originally designed for point-to-point communication patterns (so-called *unicast* communication), there is an increasing demand for using today's computer networks for group-based applications. In contrast to traditional unicast applications, group-based applications require that messages are delivered from a source to a set of receivers (i.e., the group) as opposed to a single receiver. The developing communication technology that supports group-based applications is known as *multicast*.

Networks that have been extended to support multipoint message delivery service are known as *multicast networks*. The internet protocol (IP) and asynchronous transfer mode (ATM) standards are an important class of multicast networks since they are the dominant technology deployed in public and private computer networks today. Production IP and ATM multicast networks, however, provide what is called *best-effort* multicast service for data applications. This means the network does not guarantee message delivery, ordering, throughput, or transmission delay. As a result, end-system protocols (known as *multicast transport protocols*) are commonly used to provide group transport services such as

error recovery and message ordering. Together, multicast networks and multicast transport protocols efficiently support and enable group-based applications.

Error control and receiver feedback services are a fundamental requirement of many group-based applications. For example, file and data distribution applications require in-order, complete delivery of messages from the source(s) to each member in the group. For such reliable applications, the transport protocol must detect and recover messages dropped by the network as well as inform the source when each receiver has successfully obtained the messages that compose the file. Other applications have less strict error control requirements. For example, an application that employs a replicated remote procedure call (RPC) may require that only  $k$  of  $N$  group members receive a message (so-called,  $K - reliable$  multicast). Finally, some applications classes, such as multimedia systems, do not require any error control service. Instead, the transport protocol must provide timely and accurate network drop and delay feedback so the source can determine an appropriate transmission rate. The goal of the transport protocol is to provide a rich set of error control and receiver feedback services, thereby facilitating the development of group-based applications.

Most research in multicast transport protocol design has focused on small groups (i.e., less than a hundred members) in local area networks. However, as the number of networked computer systems continues to grow, so will the demand for using multicast networks for group-based applications with hundreds or thousands of members. This domain is called *large-scale multicast*. Aside from the increased group size, large-scale multicast is a fundamentally different problem domain than local multicast because (1) network drop, delay, and throughput characteristics often vary significantly between each station and (2) wide-area networks are often saturated, resulting in highly variable drop and delay patterns. These network characteristics pose difficult challenges for large-scale error control and receiver feedback protocol designs. None the less, efficient large-scale transport protocols are

## 1.1. Background: Network Support for Multicast 3

critical because poor designs can overwhelm network links with protocol traffic and yield poor application performance.

This dissertation considers the design of efficient and low-latency error control and receiver feedback services for distributed, large-scale multicast groups. We show that transport protocols designed for small groups, in local network environments, do not scale to large, wide-area multicast groups. We present a new multicast transport framework (MESH) designed specifically for the large-scale environment. Using the MESH framework, we design a reliable data distribution protocol (MESH-R) and a deadline-driven multicast transport protocol (MESH-M). To assess the performance and network overhead, MESH-R and MESH-M are implemented in a high fidelity simulation environment based on the SURAnet and vBNSnet network architectures. We compare MESH's performance to existing classes of transport designs (namely, centralized, tree-based, unstructured, and FEC-based schemes), and show that the MESH framework compares favorably.

The remainder of this chapter motivates our research and is structured as follows: Section 1.1 presents multicast network service models provided by IP and ATM networks. We observe that both the Internet and ATM multicast data service models are inadequate for many group-based applications. Section 1.2 discusses requirements of group-based applications and observes there is a common need for error control and receiver feedback transport service. Section 1.3 gives an overview of multicast transport approaches and summarizes the inadequacies of their designs for large-scale multicast. Section 1.4 presents the form of our solution and thesis statement. Finally, Section 1.5 outlines the dissertation.

### 1.1 Background: Network Support for Multicast

When two or more networked computers exchange messages, there are many communication protocols that work together to make message delivery successful. Today's protocol architecture is organized under the OSI reference model into the following layers: physical, data link, network, and transport.



## 1.1. Background: Network Support for Multicast 4

In a layered multicast communications architecture, the data link, network, and transport protocols must support multicast service, otherwise efficient and high performance group communication can not be realized. Data link and network multicast are discussed below. Transport multicast is discussed in Section 1.2.

### 1.1.1 Data Link Multicast

Data link protocols control communication between network devices on the same local area network (LAN) segment. Thus, data link multicast provides support for multicast groups that are located on the same LAN, or LANs connected via bridges, repeaters, and switches. Fortunately, data link multicast is easily accomplished for broadcast (e.g., Ethernet) or token passing link protocols (e.g., Token-ring and FDDI) since network interface cards (NIC) can be configured to filter on both multicast and unicast frames. Today, every widely deployed data-link protocol supports multicast.

### 1.1.2 Network Layer Multicast

The network layer controls the exchange of messages between multiple LAN segments. It is critical that the network layer support multicast, otherwise a source would potentially have to unicast  $N - 1$  messages; one to each group member. With network multicast, a source simply sends a message on the multicast group address. The network elements (e.g., routers and switches) forward the message only on those links that ultimately lead to a receiver.

The Internet protocol (IP) and asynchronous transfer mode (ATM) are the two dominant network layer protocols in today's infrastructure. Their service models are discussed below:

- IP [28] was originally designed to provide best-effort datagram service with no guarantees on delay, delay jitter, or drop rate. Researchers in the Internet Engineering Task Force (IETF) are currently extending the IP protocol suite to offer QoS guarantees [19, 27]. QoS classes under development include *guaranteed service* – which

### 1.1. Background: Network Support for Multicast 5

provides hard guarantees on delay and drop rates, and *predictive service* – which provides predictive delay and drop rates.

- ATM networks offer five service classes [2, 5]: (1) constant bit rate (CBR) – which provides hard bounds on delay and drop rates, (2) real-time variable bit rate (RT-VBR) – which provides hard guarantees on delay and drop for real-time, bursty traffic such as compressed video, (3) non-real-time variable bit rate (nrt-VBR) – which provides guarantees on average delay and maximum loss rate, (4) available bit rate (ABR) – which provides minimum throughput guarantees and, (5) unspecified bit rate (UBR) – which provides no guarantees on delay, delay jitter, or drop rates.

IP and ATM service falls into two categories: best-effort (i.e., IP datagram and UBR service) and QoS (i.e., IP guaranteed, IP predictive, ABR, CBR, VBR, and rt-VBR service). The best-effort service class is intended for use by traditional data applications such as file transfer, email, remote login, transactions, and RPC. In contrast, the QoS service classes are intended to provide service for applications with time-sensitive or other media-specific constraints (e.g., a production videocast system requiring fixed bandwidth and bounded transmission delay.)

This research focuses exclusively on best-effort networks. Although QoS networks are important, they lack standardization and are not a standard service in public networks today. Adding network multicast functionality to best-effort networks requires the following three extensions: (1) multicast routing, (2) group membership detection, and (3) group network addressing. Multicast routing involves building a routing tree from source(s) to the receiver set, and dynamically modifying the routing tree to bypass congested or failed links. Group membership protocols detect dynamic membership changes (i.e., joins, leaves and failures) and inform the routing protocol to expand and prune the distribution tree as appropriate. Finally, the addressing protocol assigns the multicast distribution tree a network address (referred to as the *multicast group address*).

## 1.2. Background: Transport Support for Multicast 6

Research in multicast routing [43, 54, 76, 88, 93, 98, 100, 105], group management, and addressing is extensive. Today, IP vendors have successfully integrated multicast routing (e.g., M-OSPF [72], PIM [33], and DVMRP [34, 35, 97]), group membership (e.g., IGMP [32]), and multicast addressing into their routing and switching products. Likewise, ATM vendors have incorporated multicast routing and group management mechanisms into their signaling protocol. Although interoperability remains an issue, best-effort multicast service is available in limited form in today's production IP and ATM networks.

## 1.2 Background: Transport Support for Multicast

The transport layer bridges the gap between the application's communications requirements and the service provided by the network. In this research, we use the standard IP multicast service model given below:

- The network delivers messages to all, some, or none of the group members.
- The network does not identify which group members receive data messages, or what end-systems are listening on the *multicast group address*.
- The network does not provide feedback concerning congestion, available throughput, drop rates, or delay to the group members.

In best-effort multicast networks, two common transport protocol classes are fully reliable and multimedia transport protocols. Each is discussed below:

- *Fully reliable*: Many multicast transport protocols guarantee that a message sent by a group member is received by every group member. Thus, the transport protocol recovers messages dropped by the network, and confirms that each group member received all messages. Reliable transport protocols are typically used by distributed processing systems, data distribution systems, replicated remote procedure calls, groupware,

## 1.2. Background: Transport Support for Multicast 7

and fault-tolerant process groups. Further, reliable protocols provide service for distributed consistency protocols (e.g., a total or causal ordering and atomic delivery protocols) used by replicated filesystems, distributed databases, distributed simulations, concurrent processing systems, and cache coherence protocols.

- *Multimedia transport service:* With recent advances in network technology, it is now possible to use multicast networks for high bandwidth multimedia applications such as audiocast [50, 90], videocast [39], group video conferencing [42, 99], distributed whiteboards [41], interactive TV, and other multimedia systems (e.g., tele-medicine systems). In contrast to reliable multicast, multimedia applications are loosely synchronized groups that require real-time<sup>1</sup> data delivery, but not necessarily reliable delivery. Nonetheless, real-time applications perform best when data are not lost, and thus a transport protocol that provides *deadline-driven reliability* is warranted. Additionally, network performance feedback is useful to drive the source's coding scheme.

In addition to error control, multicast transport protocols also provide flow and congestion control as well as application-specific services such as group management, fault tolerance, and ordering services. Flow and congestion control mechanisms control the source's transmission rate and pattern such that network and end-systems resources are used efficiently and are not overwhelmed. The transport layer can also provide fault tolerance mechanisms to detect when a group member has failed, and to provide a means for members to recover missing data upon restart. Group management services handle dynamic membership changes (i.e., joining and leaving) and collect member performance data. For example, a multicast source may require throughput feedback for each member in the multicast group. Performance data allows a source to eliminate slow or unstable members of the group.

---

<sup>1</sup>Here, real-time is defined as data whose value has a limited lifetime.

### 1.3 Introduction to Previous Work in Multicast Transport

The first generation of multicast transport protocols rely on a central site for providing multicast service. Chang and Maxemchuk [21] (CM) presented an early transport protocol using a token-based approach. The token circulates around the group and (1) allows a group member to send data, (2) informs group members of messages sent in the system, and (3) provides a total message ordering. There are a number of variants of the CM approach [10, 18, 44, 53, 67, 101] designed for specific applications and network environments. Ease of implementation and simplicity are the key advantages of centralized approaches.

The drawback to centralized schemes is that network and processing overhead grows proportionally to the group size. Eventually, links close to the source become saturated, thereby limiting data throughput. The scalability limitations have motivated researchers to consider protocols that distributed protocol processing among group members. One common approach is to hierarchically organize the multicast group into a tree-based structure [47, 62, 103], thereby localizing error control and aggregating feedback. The performance of the tree-based approaches hinges on the ability to construct control trees that follow the underlying routing tree. Constructing such trees at the transport layer remains an open problem. Instead, tree-based protocols rely on manual organization (e.g., offline by an administrator), network layer routing tables, non-standard network diagnostic options (e.g., scoping and traceroute), or modify network routers to construct the control tree.

To avoid the problems associated with establishing a control tree, many large-scale protocols disseminate control information to the entire group. Timer-based multicast channel access strategies suppress duplicate control messages, while maintaining low service latency. The advantage of the unstructured approach is that there are no points of failure or “structure” to maintain. Further, unstructured approaches can be implemented entirely at the transport layer. The disadvantage is the significant network overhead incurred by multicasting control information to the group. Often receiver feedback and error control services are sacrificed to control overhead.

## 1.4 Our Approach: MESH

This dissertation develops a novel framework, called *MESH*, for large-scale multicast transport. MESH has the following characteristics: (1) it is a fully distributed, transport-layer solution, (2) it presents a robust state synchronization protocol that provides detailed end-system state for reliability, congestion control, group management, and other end-system services, and (3) it achieves efficient, low-latency error control service using a self-organizing, soft-state recovery structure (hence, the name MESH). Key novel ideas in MESH include:

- Domain-based control structure: MESH constructs a hierarchical control structure based on network domain boundaries found in networks today (e.g., local, campus, and backbone domains).
- State synchronization: MESH uses a data-driven, multicast scheme to synchronize group state. Network efficiency is achieved using special nodes, called *active receivers* (ARs), which aggregate and propagate domain state throughout the hierarchy.
- Performance-based error control: Within each domain, MESH uses sophisticated heuristics to localize recovery of dropped messages between group members. The heuristics are driven by observed network performance characteristics such as delay and drop patterns. ARs recover errors between domains.

This dissertation develops a fully reliable protocol (MESH-R), and a deadline-driven reliable protocol (MESH-M) based on the MESH framework. Using a high-fidelity network simulator, we show MESH-R and MESH-M compare favorably to (1) the tree-based, centralized, and unstructured protocol designs for single-source reliable applications, and (2) forward error correction for multimedia applications. This leads us to our thesis statement:

*Domain-based subgrouping and performance-based error control provides scalable multicast transport service with higher performance and lower network overhead than centralized, tree-based, and unstructured approaches.*

## 1.5 Structure of Dissertation

The remainder of this dissertation is organized as follows:

Chapter 2 presents previous work in multicast transport protocol designs. We consider background work in centralized, tree-based, and unstructured approaches, and we present the MESH framework in detail.

Chapter 3 develops a high-fidelity simulation environment that evaluates the performance and network overhead of multicast transport protocol designs. First we develop network models based on SURAnet and vBNSnet infrastructure. Next, we present a statistical characterization of campus and wide-area network traffic based on data collected in a backbone network. We then develop a traffic model, called  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$ , that efficiently generates background traffic with the same statistical properties found in the characterization study. Finally, we derive  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  parameters to drive the background load in the SURAnet and vBNSnet network simulations.

Chapter 4 develops MESH-R, a reliable multicast transport protocol based on the MESH framework. We implement MESH-R and three other protocols (namely, centralized, tree-based, and unstructured protocols) in the SURAnet and vBNSnet simulation environment presented in Chapter 3. We compare the performance and overhead of each protocol for a single-source, bulk file distribution application. Our analysis demonstrates the effectiveness of MESH's state synchronization and error control protocol. We show that MESH compares favorably to the other protocol designs across a range of network utilization.

Chapter 5 develops MESH-M, a multicast transport protocol that provides deadline-driven reliability suitable for multimedia applications. We compare the performance and overhead of MESH-M to a number of FEC schemes in the SURAnet environment. Our analysis shows that with some additional playback delay, MESH-M is attractive as compared to FEC schemes across a range of network loss patterns.

Chapter 6 presents our conclusions, summarizes the contributions of this dissertation, and outlines future research directions.

## Multicast Transport Protocols: Previous Work

---

Contemporary multicast transport protocols can be characterized based on the type of services provided, network model assumptions, target group size, and number of sources supported (see [11, 40] for a complete characterization). Transport services are the most discriminating of these characteristics, and can be roughly divided into three classes: (1) ordering protocols, (2) fully reliable protocols, and (3) real-time protocols. Sections 2.1, 2.2, and 2.3 discuss previous work for each service class. Section 2.4 discusses current research efforts aimed at improving the scalability of multicast transport protocols. Finally, Section 2.5, presents the MESH framework.

### 2.1 Multicast Ordering Protocols

Early research in multicast transport protocols focused on providing reliable, atomic, and ordered message delivery for  $N \times N$  distributed systems<sup>1</sup>. In addition to providing traditional end-to-end services, multicast ordering protocols ensure that messages are delivered according to the ordering constraints specified by the service model or application (e.g., causal or total ordering).

---

<sup>1</sup>In  $N \times N$  protocols every group member is potentially a source.



## 2.1. Multicast Ordering Protocols 12

Chang and Mexemchuck (CM) [21] present the pioneer work on reliable, ordered multicast protocols. The CM algorithm circulates a token to all group members. The site holding the token multicasts a sequenced acknowledgment for each message sent to the group, thereby creating a total order. Token rotation is reliable; thus, failed members can be detected.

There are many extensions and variations of the CM approach. Whetten [101] presents the Reliable Multicast Protocol (RMP) which allows multiple groups, multiple ordering constraints (i.e., source ordering and/or total ordering,) and multiple reliability guarantees (i.e., reliable, unreliable or  $K$  reliable). Armstrong et al. [10] presents the multicast transport protocol (MTP) which further centralizes the CM algorithm by electing a “master” site to handle error control, flow control, ordering, and membership. All group members register with the “master” site as either a producer and/or consumer. The master schedules the producers such that only one producer is active at any given time, thereby guaranteeing a total ordering. Bormann et al. [18] propose MTP-2 which extends MTP’s group joining procedure, recovers if the master fails, allows the master to migrate, and prioritizes the producers. Kaashoek et al. [53] presents RBP in which all messages are unicast to a central site called the sequencer node. The sequencer, in turn, orders and multicasts the message to the group. Spauster [44] extends the CM approach to multiple groups. Each subgroup is assigned a central sequencing node, called a primary, which sends and receives messages for the subgroup. Primaries exchange messages such that subgroup streams are combined in a consistent manner as they are routed to their destination, thereby creating a total ordering. Maffies [67] also studies total ordering among multiple groups. Each subgroup is assigned a group transport service (GTS) agent, which provides a total ordering within the subgroup (inter-group total ordering is not addressed).

There are many other transport ordering protocols [14, 16, 15, 26, 55, 69, 73, 83] which are not discussed here since their primary focus is the ordering algorithm, as opposed to receiver feedback and error control.

## 2.2 Reliable Multicast Protocols

The drawback to the multicast ordering protocols presented in Section 2.1 is the latency and network overhead incurred by the ordering algorithm. For an application that requires reliable multicast, but no multiple source ordering services, more efficient and lightweight protocols are sufficient.

The Xpress Transport Protocol (XTPv4) [4, 95] provides multicast service primitives that offer reliable and unreliable service for both single and multiple source groups. A central agent, called the multicast group manager (MGM), maintains a list of group members and traffic specifications used by the error, flow, and congestion control algorithms. The application can specify which, if any, of the group members must receive the data reliably, as well as how to handle member failures. For multiple sources, XTP is similar to RBP in that each source first unicasts messages to the MGM, which in turn multicasts the message to the rest of the group. The communication between the sources and the MGM is a normal XTP unicast connection; thus, the MGM manages rate, flow, and error control independently for each source via XTP unicast. To desynchronize and reduce the amount of return traffic to the MGM, XTP uses slotting, damping, and polling techniques. Slotting is a technique whereby receivers randomly back-off retransmission requests. By multicasting retransmission requests other receivers can “damp” out redundant requests.

Jain and Ramakrishnan [86] optimize the basic centralized approach by incorporating negative acknowledgments and periodic polling messages (NAPP) for single-source reliable multicast in LANs. Like XTP, error control uses negative acknowledgments (NACKs), slotting, and damping. To ensure forward progress of the sender, each receiver periodically sends a positive acknowledgment (ACK). However, ACKs are only required if there are no errors (i.e., NACKs implicitly acknowledge correctly received messages.)

Crowcroft [30] and Cheriton [23, 24, 22] study LAN-based client-server applications<sup>2</sup>. In Crowcroft’s protocol, the client maintains send and receive windows for each server in

---

<sup>2</sup>In the client-server domain, the servers make up the multicast group.

the group. The minimum of the send windows establishes the amount of data the client can send to all servers, and the individual server windows limit the retransmissions to any given server. Error control is achieved using positive acknowledgments. Thus, clients can desynchronize return traffic flow from the servers by delaying acknowledgments. In contrast, Cheriton's VMTP protocol uses the response from the server as an implicit acknowledgment. If the server response or client request is lost, the client will time-out and resend the request. VMTP does not address flow and congestion control.

Ammar [9, 25] improves the latency of single source, bulk data distribution by "parallelizing" the multicast stream. The protocol partitions the receiver set into subgroups based on throughput and delay characteristics from the source. Thus, members with high throughput will be grouped together and will receive the data in much less time than the slow subgroups. Ammar also proposes the single connection emulation protocol (SCE) [96] which lies between IP and TCP. The SCE layer handles group management and aggregates SCE acknowledgments into a single TCP ACK. If all acknowledgments are not received by the SCE protocol, TCP will time out and send a retransmission.

As opposed to retransmission, many protocols over-code the data stream such that many network losses can be recovered without using retransmission. Jones [52] presents a data distribution protocol based on saturation (also referred to as replication) combined with negative acknowledgments for error control. More recently, Nonnenmacher [74, 75] combines a forward error correction scheme [13, 87] (FEC) with centralized retransmission to achieve reliable data distribution. The results show the FEC/ARQ hybrid error control protocol (called NP), offers a substantial reduction in network overhead compared to pure retransmission schemes.

## **2.3 Real-Time Multicast Protocols**

For optimal performance, real-time applications require bounded transmission delay, error rates, jitter, and throughput guarantees which can only come from the multicast network.

Such QoS guarantees obviate the need for transport-level error and congestion control. Since QoS networks are not publicly available today, many real-time applications rely on data-grade service found networks such as the MBONE. Real-time transport protocols in best-effort multicast networks (or probabilistic and predictive QoS networks that exhibit dynamic variations in QoS) primarily focus on network performance monitoring, group membership, and other feedback service.

The real-time protocol (RTP) [91, 92] provides session and transport services for multi-party interactive multimedia conferences<sup>3</sup>. RTP transport services include QoS and receiver-set performance monitoring. In RTP, each member periodically multicasts reception performance and sending state information. The sending state component includes the source identifier, NTP timestamp [70, 71], RTP timestamp, packet count, and octet count. The reception component contains performance reports for up to 31 senders. Each report includes the source's identifier, cumulative packets received, cumulative packets expected, inter-arrival jitter, and the timestamp of the last report.

Yavatkar [104] investigates rate-adjustment, error control, and feedback strategies for large-scale distribution of time-sensitive data. The following rate adjustment strategies are evaluated: (1) majority policy - the rate is set to the majority of the receivers, (2) universal policy - the rate is set to the minimum of the receivers, and (3) probabilistic policy - the rate is probabilistically increased or decreased based on receiver feedback. Three redundant transmission strategies are evaluated: (1) replication - a packet is duplicated every  $K$  packets, (2) X-OR - an X-OR packet is generated every  $K$  packets, and (3) replication and X-OR combined. Finally, three receiver feedback strategies are evaluated: (1) immediate feedback, (2) random back-off feedback, and (3) random back-off with probabilistic feedback. Based on simulation experiments, the authors conclude probabilistic rate reduction, replication and random feedback back-off are the most promising techniques.

---

<sup>3</sup>The full set of services include: synchronization, stream multiplexing (mixing), stream demultiplexing, media encoding identification, error detection, encryption, and QoS monitoring.

### 2.3. Real-Time Multicast Protocols 16

Bolot et al. [17] propose a scalable feedback mechanism for single-source, large scale videocasting in multicast data networks. The video source uses the receiver feedback to adjust the video data rate to match the available bandwidth in the network. Feedback is accomplished using sender-based polling with key-based matching. Both the sender and receiver randomly generate keys at the beginning of a polling period. Sender polling messages include its key and a bitmask. If the receiver's key matches the sender key under the bitmask, then it responds with its state (i.e., loaded, unloaded or congested). Once a receiver replies for a given polling period, it does not respond to any other polling requests. By increasing the number of significant bits in the mask the sender can control implosion, yet receive a response from every group member.

Braudes and Zabele present the reliable and adaptive multicast protocol (RAMP) [20] for real time distribution of images. RAMP assumes a network with QoS capabilities (such as the RSVP [106] approach proposed for IP) and supports priority data. The priority information is kept within the IP header, thus RAMP requires router support. Error control is achieved using negative acknowledgments with selective retransmission. Flow and congestion control is achieved using a dynamic rate-based mechanism driven by source quench messages from routers and error rates.

In addition to feedback service, recent research focuses on layered distribution of real-time data. McCanne [68] presents the receiver-driven layered multicast (RLM) protocol for distribution of hierarchically encoded [93] video streams. RLM uses a parallel approach similar to Ammar's scheme [9, 25], except each band in the video stream is multicast on a different address. Thus, slow receivers tune into low-frequency image bands, whereas receivers with a fast connection tune into all bands thereby receiving the highest quality image. Li [59, 60] presents the layered video multicast with retransmissions (LVRM) protocol which extends RLM with retransmission-based error control and hierarchical rate control.

## 2.4 Distributed Multicast Transport Protocols

The protocols discussed in Sections 2.1, 2.2, and 2.3 rely on a central site to keep track of all the receivers in the group; we call this the *centralized approach*. Central sites greatly simplify many of the end-to-end, group management, and ordering algorithms. However, one serious problem is that the central site must process all the control messages and protocol overhead. Therefore, the performance of the multicast session is bounded by the capacity of the central site to process protocol messages and receiver state. Eventually, as the group grows in size, the protocol overhead either exceeds the central site's processing capabilities or exceeds the available bandwidth of the communications links at the central site (i.e., multicast implosion).

The protocols reviewed either do not consider scalability (i.e., were designed for small groups), or use mechanisms to reduce the number of messages between the central site and receiver set. For large groups, Pingali [84] shows that receiver-initiated approaches (e.g., negative acknowledgments, slotting, and damping) are effective control message reduction techniques. However, receiver-based approaches perform best when receivers “eavesdrop” on control messages sent from other receivers. Thus, control messages must be sent on the multicast address (which is inefficient in terms of bandwidth utilization), or all receivers must be located on a broadcast network (which is not the case for large groups). Further, receiver-based approaches are not suitable for applications in which the source must positively hear from the receiver set on a per message or periodic basis. Many protocols address this problem using probabilistic polling and other polling acknowledgment desynchronization techniques. These techniques work well for applications which can tolerate large polling periods; however, acknowledgment desynchronization schemes are inappropriate for large groups which require low latent, tight synchronization.

In general, control message desynchronization, polling, NACKs, slotting, and damping techniques improve protocol scalability for certain applications and network environments. However, as the receiver set grows the central site must ultimately sacrifice performance by

quenching the data traffic or sacrifice services such as error recovery, group management, and performance feedback. These scalability limitations are driving researchers to consider distributed transport protocol design for large-scale, wide-area multicast.

### 2.4.1 Distributed Approach at the Network Layer

One distributed technique is to push error control and state-feedback services into the network layer. For example, Rajapolan [85] presents the reliable multicast (RM) protocol which distributes protocol processing into the network layer to achieve reliable, K-reliable, and subgroup-reliable multicast. Error recovery is local between router pairs, thereby incurring low network overhead and recovery delay. Each channel between router pairs uses window-based flow control, which in turn regulates the amount of data each source can send. A *gather* operation allows sources to efficiently obtain state information from the group. Papadopoulos [77] proposes embedding retransmission service into network routers known as *subcast*. Routers are configured to accept retransmission requests and replies from end-systems, and forward them onto links which lead to receivers missing the data. Cisco [94] has developed a similar scheme to Papadopoulos, except end-systems send NACKs directly to the source (as opposed to a router). PGM enabled routers intercept NACKs, and subcast an ACK to suppress redundant NACKs. If an end-system receives a router ACK and has a copy of the data, it unicasts the retransmission to the source. PGM routers intercept retransmissions and subcast it to the group.

Embedding error control and receiver feedback aggregation into the network layer is an optimal solution because, (1) only a single request and reply are generated on a single link, (2) retransmissions only traverse links which lead to a receiver missing the data, and (3) requests and replies travel the routing tree. The drawback to RM, PGM, and Papadopoulos's scheme is that network-layer protocols, software, and hardware must be modified to support the services. Although this approach may lead to high performance and low cost reliable multicast, we believe it is unrealistic to expect these mechanisms to

be a standard service within network protocols such as IP, or link layer technologies such as ATM and frame-relay.

### 2.4.2 Tree-Based Protocols

Paul [78] was the first to introduce a distributed transport protocol architecture, called the Designated Status Protocol (DSG). DSG is similar to the RM protocol, except designated end-system receivers (DRs) perform error recovery and aggregate acknowledgments as opposed to network routers. Like the RM approach, DRs enable error recovery and control packet aggregation to proceed independently and concurrently in different parts of the network.

Figure 2.1 shows how DRs might be hierarchically organized in the SURAnet [3]<sup>4</sup> network architecture. The dark dashed lines indicate the multicast routing tree<sup>5</sup> for a source located at CTV and receivers located at each campus. Inside an arbitrary campus, say GIT, communication is fast, cheap, and predominantly error-free. Thus, receivers inside GIT will receive nearly identical subsets of the packets sent on the multicast address. Thus, it makes sense to establish a DR representing the GIT campus. Further, campuses “downstream” from GIT in the multicast routing tree (e.g., TAU, JKV) receive a subset of the messages the GIT campus receives. Thus, it makes sense that the GIT DR is a parent node of TAU and JKV. For example, Figure 2.1 shows receivers at JKV, MSB, and AUB sending protocol control messages to the JKV DR. In turn, the JKV DR sends aggregated protocol control messages to the DR located at GIT.

Lin and Paul [62] present the reliable multicast transport protocol (RMTP) which is based on the RM protocol and the DSG architecture to provide reliable, single-source, data distribution service. RMTP uses windowed flow control with congestion avoidance [48] to prevent overwhelming end-systems and network resources. Therefore, RMTP adapts the sending rate to the slowest receiver. Receivers and DRs use a timer-based approach to

---

<sup>4</sup>SURAnet is a contemporary WAN connecting research institutions in the southeastern United States.

<sup>5</sup>A source-based tree, in this case.



## 2.4. Distributed Multicast Transport Protocols 20

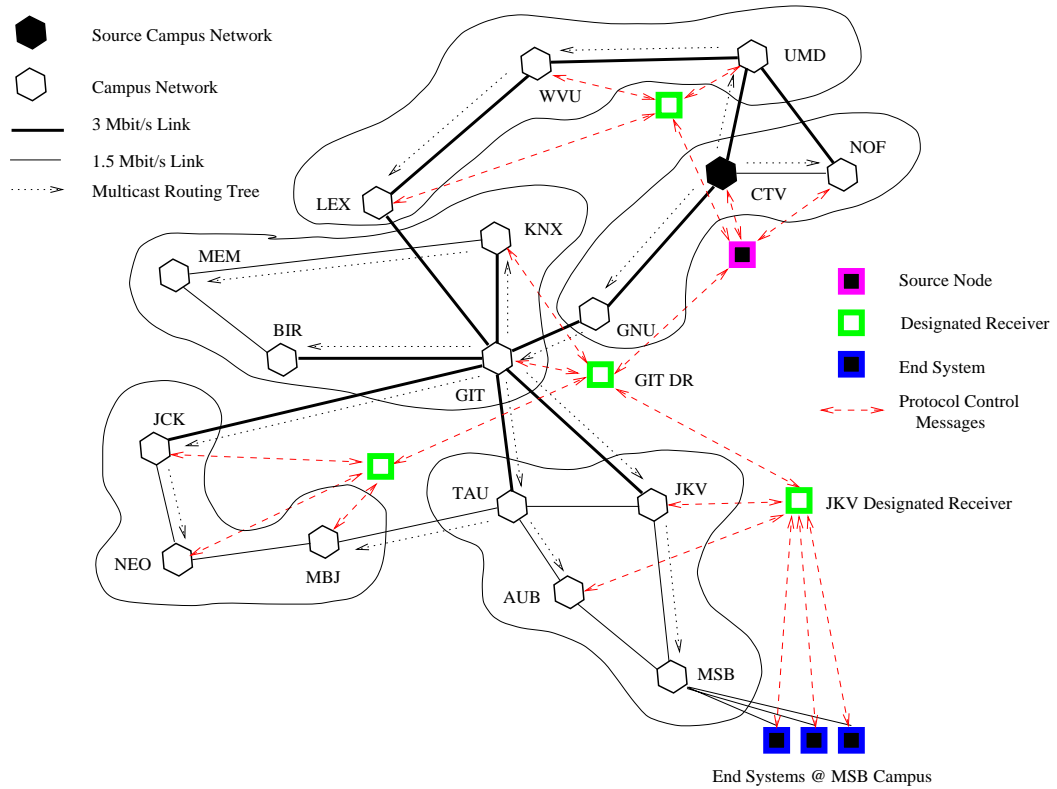


Figure 2.1: Subgrouping Example of SURAnet

periodically unicast ACKs up the DR hierarchy. ACKs include the largest sequence number  $L$  of the message in which all subsequent messages have been received, a bitmap indicating which messages were lost after message  $L$ , the receiver's maximum window size and the maximum data rate the receiver can process. The source and DRs use ACKs to retransmit data, adjust sending rates, release buffers and clean up aged state information. DRs cache all the data sent on the multicast address; thus, a receiver can join late or recover from failure and still receive the complete data stream. When a DR sends a retransmission via multicast, it only sends the message on the subtree. Thus, like RM, RMTP must modify network routers and routing protocols.

## 2.4. Distributed Multicast Transport Protocols 21

Holbrook et al. [47] present the log-based receiver-reliable multicast protocol (LBRM) for distributed interactive simulations. LBRM uses a receiver-reliable approach to error control but does not consider fault tolerance, flow, rate or congestion control (these are considered application-layer issues). LBRM is similar to RMTP in that designated receivers<sup>6</sup> distribute error-recovery overhead and log all messages sent from the source. If the receiver detects a gap in the sequence space and wants to recover the lost message, it unicasts a retransmission request (i.e., NACK) to the nearest logging server. Based on the number of NACKs, the logger can decide to unicast or multicast the retransmission. In contrast to RMTP and the polling mechanisms found in NAPP and XTP, LBRM uses a heartbeat scheme to provide sub-second loss detection of a single packet. Whenever a message is transmitted, the source includes the maximum time (i.e., heartbeat) before it will send another message. If the source does not have data to send after the heartbeat has expired, then it sends a heartbeat message with a larger heartbeat interval. LBRM is a strict transport layer solution. The difficulty is that logging servers must be strategically placed in the network.

Yavatkar [103] presents the tree-based multicast transport protocol(TMTP) for single-source data distribution. TMTP is a distributed protocol based on subgrouping and DRs, thus it is very similar to RMTP. Like RMTP, each DR periodically sends positive acknowledgments to its parent in the subtree for flow control. Unlike RMTP, TMTP receives multicast NACKs using IP's TTL scope control<sup>7</sup>. This enables other receivers to damp out redundant retransmission requests. The local DR also retransmits the packet under TTL scope control such that all local receivers recover the message. Flow control is achieved using a combination of window and rate-based control mechanisms. The rate is established at the beginning of the session, and the windows are updated at each DR from ACK control messages.

---

<sup>6</sup>Designated receivers are called *logging servers* in the LBRM protocol.

<sup>7</sup>Time to live (TTL) is value carried in the IP packet. Each time a router handles a packet, it decrements the TTL field by one. If a router receives a packet with a TTL of 0, the router discards the packet.

### 2.4.3 Unstructured Protocols

Tree-based protocols perform exceptionally well when the control tree maps to the underlying network routing tree. However, constructing such trees at the network layer is still an open problem. Alternatively, some protocols do not create an explicit control structure, and multicast control messages to the entire group. Floyd et al. [41, 49] use an unstructured approach called scalable reliable multicast (SRM) to provide transport service for Wb - a shared, interactive whiteboard tool. Receiver feedback is accomplished by each member periodically multicasting session messages<sup>8</sup>. Session messages contain the highest sequence number received from every source in the group. When a node detects the network has dropped a message, it multicasts a retransmission requests to the group. To avoid message implosion, the request is slotted and redundant requests are damped. The closest group member which has the data multicasts the retransmission; again, other receivers damp out redundant retransmissions. Since SRM is unstructured and purely distributed, it has a significant robustness and portability advantage over RTMP, TMTP, and LBRM. However, the cost and delay of multicasting error and session messages is substantial.

## 2.5 The MESH Framework

We present a general framework suitable for building large-scale transport services in [63]. Known as *MESH*, the framework supports rich state feedback and error control services for single and multi-source reliable, multimedia, and other multicast application types. Key criteria motivating the design of MESH include:

- Distributed overhead: Control processing and message overhead must be distributed throughout the multicast group for a protocol to scale beyond a few dozen nodes or to wide-area networks.

---

<sup>8</sup>The rate of session messages is targeted to be 5% of the target bandwidth of the Wb session.

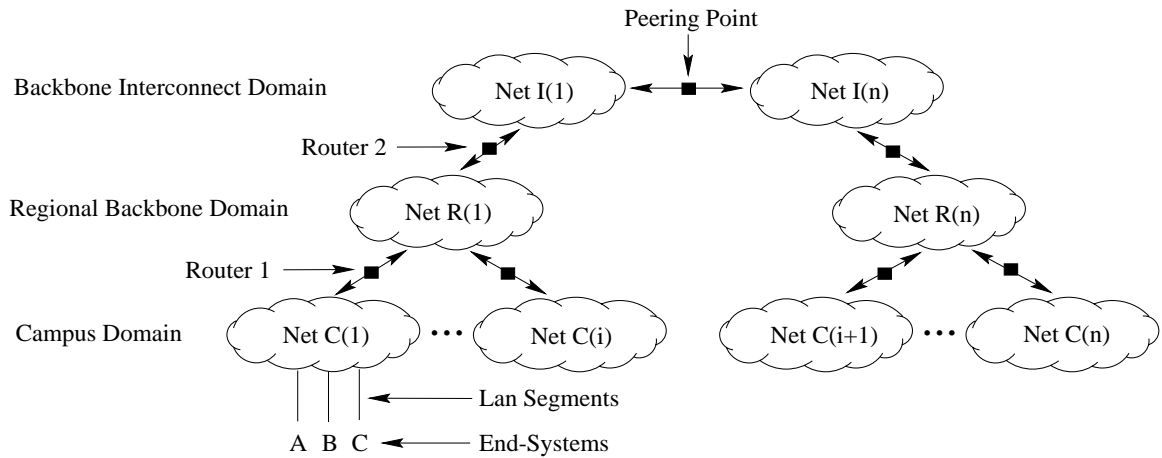


Figure 2.2: Internet domain hierarchy

- **Localized service:** Protocol services such as error control and receiver feedback must be localized to achieve efficient, low-latency service.
- **Network independence:** Protocols must not rely on network-level support beyond basic IP-datagram service, otherwise they will not port to other network environments.
- **Dynamic control structure:** Protocol control structures must adapt quickly to dynamic network congestion patterns, changes in group membership, and gracefully handle failures.

### 2.5.1 MESH Domains

MESH partitions the multicast group into subgroups based on network domain boundaries. For example, consider Figure 2.2 which shows the hierarchical organization of the Internet today. The figure shows four network domains: local, campus, regional, and backbone interconnect. The lowest level (LAN segments) connects end-system device (e.g., *A*, *B*, and *C*), whereas the highest level (backbone interconnect) provides long-haul, wide-area transit.

MESH’s domain-based subgrouping strategy is motivated by (1) infrastructure found in each network domain, (2) the nature of packet losses and transmission delays in hierarchically organized networks, and (3) the ability to domain-scope multicast messages. In particular, MESH’s design is based on the following observations:

- Bandwidth is plentiful in the local and campus area networks, and backbone networks represent the constrained resource.
- Packet loss and delay occurs primarily at the network exchange/peering points and in the backbone networks. Thus, members within a given campus domain will receive a similar set of packets and transmission delay relative to a particular multicast source.
- Domain multicast is easily accomplished. For example, consider a node on campus network Net C(1) in Figure 2.2. In IPv6, multicast messages can be domain-scoped such that it is only delivered to the nodes attached to Net C(1). In IPv4 networks, domain scoping is also feasible via a local multicast address, or gateway service (e.g., configuring a firewall at “router 1”).

Once the multicast group is hierarchically organized based on network domains, the goal becomes localizing protocol processing as much as possible, thereby reducing service latency and network load.

### 2.5.2 MESH State Synchronization

In MESH, receiver state is first synchronized locally within the domain, then between domains. Note that exactly what state is exchanged, and the degree of synchronization depends on the application’s requirements. Nodes within a domain synchronize state using domain-multicast service. For example, consider end-system *A* on network C(1) in Figure 2.2. By domain-multicasting, *A*’s state messages are forwarded to end-systems within network C(1) (i.e., *B* and *C*.) Domain-multicast has the following desirable properties: (1)

each member can identify and track the state of other members within its domain, (2) any node can compose an aggregate state message representing all the end-systems within the domain, and (3) there is no control structure to create or dynamically maintain.

Property (2) above allows a single member to report the state of the domain. MESH leverages this property by selecting a single node within each domain to act as the domain's "active receiver" (AR). The key role of the AR is to aggregate and forward domain state to the next domain in the hierarchy. For example, an AR on Net  $C(1)$  can regionally domain-multicast a state message such that it is scoped to networks  $C(1) - C(i)$ . Thus, all receivers within network  $C(2) - C(i)$  can determine the state of the receivers in network  $C(1)$  with a single message. State synchronization is recursive; that is, campus domain ARs elect a regional domain AR to report the state of receivers located in networks  $C(1) - C(i)$ .

### 2.5.3 MESH Error Control

In the MESH framework, error control is based on a unicast request/response retransmission strategy. Unlike the tree-based approaches, MESH does not build an explicit recovery tree. Likewise, domain-multicast retransmissions are avoided due to the network overhead. Instead, MESH relies on a number of heuristics to localize error control based on RTT estimates between receivers and network error patterns observed. Like the state feedback protocol, the MESH error recovery protocol first tries to locally recover lost messages within the domain. For those packets that cannot be recovered locally, the AR attempts to recover the message in the next higher domain. Like the state synchronization protocol, MESH's error control and server selection heuristics are based on the type of service required by the application (e.g., fully reliable and deadline driven reliable) as well as the desired efficiency and latency.

## 2.6 Summary

This chapter presented previous work in multicast transport protocol design. We first considered centralized protocols which provide ordering, reliability, and feedback services for real-time data. We concluded that the centralized design does not scale to large groups or wide-area networks. Next, we considered scalable protocols that distribute control processing using a tree-based and unstructured approach. We showed that the tree-based protocols present a scalable and efficient solution, however, they require support from network software in order to localize services. We considered unstructured approaches and showed that they are a robust transport-layer solution. However, they are inefficient and incur substantial network overhead as compared to tree-based designs. Finally, we presented the MESH framework. We argued that the framework: (1) distributes network overhead using a domain-based control structure, (2) provides a rich state feedback architecture based on domain-multicast and AR state aggregation, and (3) provides localized error control via a unicast request/response approach.

## Wide-Area Multicast Network Simulation

---

This chapter develops a technique suitable to evaluate the performance and overhead of large-scale multicast transport protocol designs such as MESH. In the literature, common protocol performance evaluation approaches include analytic models, simulation, and empirical studies. Analytic evaluation is often the first technique employed because (1) theoretical performance is easily obtained when the protocol is expressed mathematically, (2) they are quick to compute, and (3) they give fair comparative performance between protocol designs. However, analytic models are not well-suited to evaluate MESH and other distributed multicast protocols because such protocols are not easily expressed mathematically. For example, researchers have yet to analytically model a modern unicast flow control algorithm, such as that found in TCP, because of the complex, dynamic interaction with the error control algorithm, congestion control algorithm, round-trip time estimators, and network state[8]. Distributed multicast transport protocols, such as MESH, represent an even greater challenge to express mathematically due to the additional dependencies between the end-systems in the group.

Empirical evaluation is in some respects ideal because the protocols are evaluated under conditions of real network congestion, link errors, transmission delays, and router drops. In this work, however, empirical evaluation is precluded because (1) extant wide-area data



networks do not offer experimental multicast service, (2) detailed network management statistics such as per-link loss and delay information are not available, and (3) there is no means to control network load. Due to the limitations of analytic and empirical techniques, computer network simulation is commonly used to evaluate large-scale transport protocols for multicast applications. In general, simulation is attractive because protocol performance and overhead can be predicted for contemporary and future high-speed networks under a range of network congestion levels.

This chapter develops an empirically driven, high-fidelity network simulation of SURAnet and vBNSnet. We show the simulation environment is suitable to: (1) measure network overhead introduced by a multicast protocol, (2) support large multicast groups with dynamically changing membership, (3) support single and multi-source distributed, multimedia, and continuous stream applications, (4) support large-scale, heterogeneous network topologies with contemporary and next-generation infrastructure, (5) control network load and congestion, and (6) run efficiently on a contemporary high-end workstation.

The remainder of this chapter is organized as follows: Section 3.1 introduces the structure of the simulation environment developed and discusses issues related to discrete event network simulation. Section 3.2 presents the local campus and wide-area networks considered. Section 3.3 presents the traffic modeling approach used to drive delays and packet drops within the campus network. Next, Sections 3.4-3.6 introduce and motivate the WAN traffic model by presenting a comprehensive characterization study of the statistical properties of a wide-area network packet arrival process. Sections 3.7-3.8 present the details of the WAN model, and evaluates its accuracy in terms of the empirical traces. Finally, Section 3.10 integrates the traffic model into the network simulation described in Section 3.2, and gives model parameters.

### 3.1 Introduction to Computer Network Simulation

Computer network simulations have two main components: the network model, and the background traffic model. The network model consists of the transmission infrastructure, networking elements (e.g., switches and routers), end-systems, and protocols which define how messages are delivered from source to destination systems. The traffic model generates background load within the network, thereby driving queuing delays, packet drops, transmission delays, and other network performance characteristics. Once the network and background traffic models are established, the protocol under consideration is implemented within the end-systems (as is done with a real system) and the simulation is executed. Using statistics gathered within the simulator, protocol overhead and application performance can be expressed quantitatively for the network, group size, congestion, and application under consideration.

In this research, we use a packet-level simulation approach to modeling computer networks. In a packet-level simulation, each network element (such as a router, transmission link, and end-system) is modeled as a distinct entity. Network packets flow individually from the host system, through the network elements, and finally to the destination. Each network element models packet flow within the device. For example, consider a network router. Packets arrive at the input port, wait in the input buffer, transfer across the backplane, and finally wait at the output port until the transmission link becomes available.

The key advantage to packet-level simulation is that with careful design, accurate network performance characteristics can be achieved. The drawback, however, is that modeling each packet and each network element requires significant computational and memory resources. For example, at a minimum, a discrete event simulator [8] must schedule two events per packet transmission: (1) the packet must be queued in a local buffer until the network link is available and (2) the packet must be scheduled to arrive at the destination node. Since each event requires a 100 – 200 byte data structure within the simulator, substantial memory is required even for a simple network architecture. Thus, packet-level simulation is

### 3.2. SURAnet and vBNSnet Network Model 30

often precluded for large networks, such as the global Internet [82]. In this research, we limit the network infrastructure, network topology, simulation timing granularity, and precision of the network element models such that a five minute network simulation executes in less than 24 hours on a Sun UltraSparc 170 equipped with 320 MB of memory.

In a packet-level simulation, the network performance characteristics (e.g., end-to-end transmission delays and drop rates) are driven by the background traffic model. Thus, it is critical that background traffic models generate realistic network loads, otherwise any protocol performance predictions will not be representative of performance found in a production network. Computational efficiency, however, is of equal importance since millions of packets per second may have to be generated in a high-speed network environment. Hence, the goal of the background traffic models developed in this chapter is that it should be efficient enough for use in a regional or national Internet backbone network simulation, while retaining sufficient accuracy to faithfully model campus and wide-area network performance characteristics.

## 3.2 SURAnet and vBNSnet Network Model

This section develops network models of SURAnet and vBNSnet – shown in Figures 3.1 and 3.2 respectively. SURAnet is chosen because it represents a large-scale, contemporary Intranet or ISP network employing low-speed, T-1 infrastructure. Whereas, vBNSnet is chosen because it represents a state-of-the-art, high-speed WAN based on 155 mbps OC-3c infrastructure. Together, the SURAnet and vBNSnet network models provide a basis to evaluate the performance and overhead of multicast protocols in low and high-speed network environments.

SURAnet and vBNSnet provide backbone interconnect service for large university, corporate, and small ISP networks. In the network model developed here, each SURAnet and vBNSnet network access point (NAP) provides service for the campus network shown in Figure 3.3. This simplified campus network is modeled after the University of Virginia net-

### 3.2. SURAnet and vBNSnet Network Model 31

work (UVAnet) which uses FDDI backbones to interconnect Ethernet LANs. As Figure 3.3 shows, there is a four route hop between end-systems which reside within the campus network (but not on the same LAN segment), and a two-hop route to the backbone network. End-systems in the multicast group can be located on any one of the three Ethernet segments. Thus, the routes between the end-systems are sufficiently distinct such that campus delays and drops create different performance characteristics for multicast group members located on the same campus network.

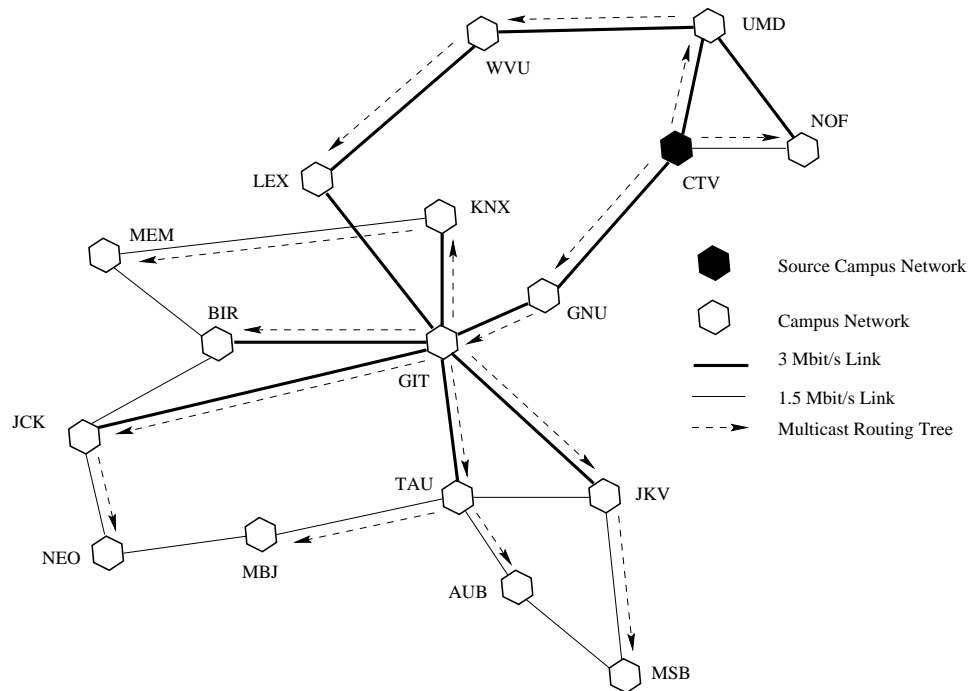


Figure 3.1: SURAnet infrastructure and topology.

We use the Internet Protocol version 4 (IPv4) [28] as the network layer protocol. The IPv4 protocol specification provides “best effort” datagram service; that is, the network makes no guarantees on throughput, transmission delay, transmission delay variation, delivery order, or drop rates. The basic IP packet consists of a source address, destination address, checksum, transport protocol identifier, packet length, transport protocol payload,

### 3.2. SURAnet and vBNSnet Network Model 32

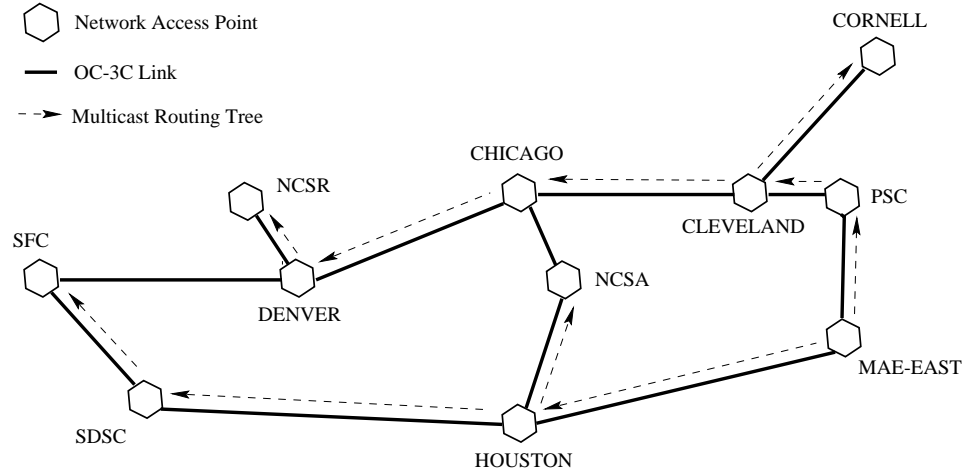


Figure 3.2: vBNS infrastructure and topology.

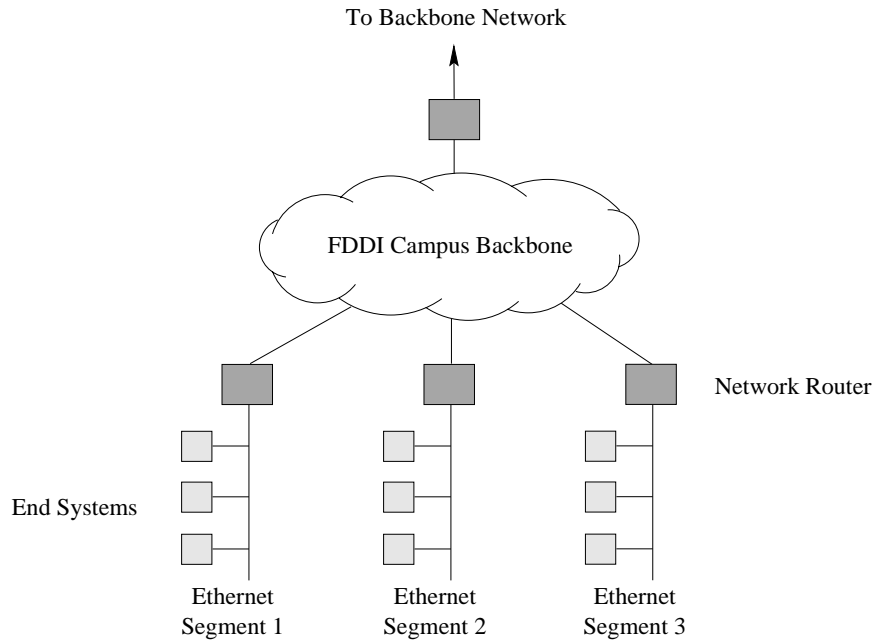


Figure 3.3: Campus network model.

### 3.2. SURAnet and vBNSnet Network Model 33

and optional diagnostic fields. The standard IPv4 header (i.e., those without options defined) is 20 bytes in length. IP packets are delivered from a source to a destination using a routed approach; thus, there is no explicit network path (i.e., connection) setup between end-systems in the group. Instead, routers use the packet's destination address as an index into a routing table which determines the output link. After determining the appropriate output port, the router queues the packet at the output interface buffer until the link becomes available.

In the Internet, routing tables are dynamically constructed using a link state (e.g., MOSPF[72]) or distance vector (e.g., DVRMP [34, 35, 97]) routing protocol. For simplicity, the SURAnet and vBNSnet network models use static, shortest path first routing which minimizes the number of hops. For example, Figure 3.1 illustrates the multicast routing tree constructed for a source located at the CTV network access point (NAP) with receivers located on each campus network. A variant of the Internet group management protocol (IGMP) [32] determines the network location of the multicast group members and detect dynamic changes. IGMP informs the routing protocol to expand and prune the routing tree as members join and leave the group.

Within the campus network, data link multicast delivers multicast packets to receivers on the same Ethernet segment. The campus network model also employs IPv6's domain-scoping service. Routers at the WAN network boundary forward packets that have the domain scope flag only to internal campus links. Although domain-scoping is not a standard service within IPv4, domain-multicast is easily accomplished today using a local multicast address.

Within the simulation, the end-to-end network delay of a packet consists of the sum of link propagation delays (fixed on each link), router queuing delays, and packet transmission times. Network routers drop a packet if there is insufficient output port buffer space or CPU cycles available using a tail-drop discard strategy.

### 3.3 Campus and Local Area Traffic Model

Background traffic is generated by two sub-models: one which generates background traffic within the campus network, and one which generates background traffic within the wide-area network. This section presents the traffic model used within the campus network. Sections 3.4-3.8 present a parameterized traffic model which generates background traffic within the WAN.

The goal of the campus network traffic model is to generate background load such that application packet delay and loss rates represent the performance characteristics of UVAnet. Figures 3.4 and 3.5 show delay and drop rates (where drops are shown as the negative spikes) for an unicast 32 kbps voice stream over two and three hop routes within UVAnet<sup>1</sup>. Of the five network paths studied (ranging from one to five hops in length), the network performed very well with average transmission delays ranging from 3 – 7ms (one-way), and an isolated drop rate of less than 1% of the stream. This data suggests that UVAnet is engineered properly to support resource-hungry multicast applications and has significantly lower delay and drop rates as compared to the Internet today.

Figure 3.6 shows a common approach to modeling delays and drops at a LAN segment router. The core idea is to statistically multiplex cross-traffic (shown as dark packets) with application packets (shown as light packets) at each router's output port. The statistical properties of the cross-traffic are carefully controlled such that the queuing delay and drop rates correspond to router delay and drop statistics measured within UVAnet.

We employ the cross traffic approach at each UVAnet campus router. We derive the target packet drop rates by considering the utilization (in both packets and bytes per second) and drop rates (due to output port buffer overflows) for the 14 Ethernet and 2 FDDI interfaces presented in Table 3.1. The table was constructed by using the simple network management protocol (SNMP) to query the Olsson router once per second during a weekday

---

<sup>1</sup>Refer to [38] for detailed analysis of transmission delay and drops as well as details concerning the experiment.

### 3.3. Campus and Local Area Traffic Model 35

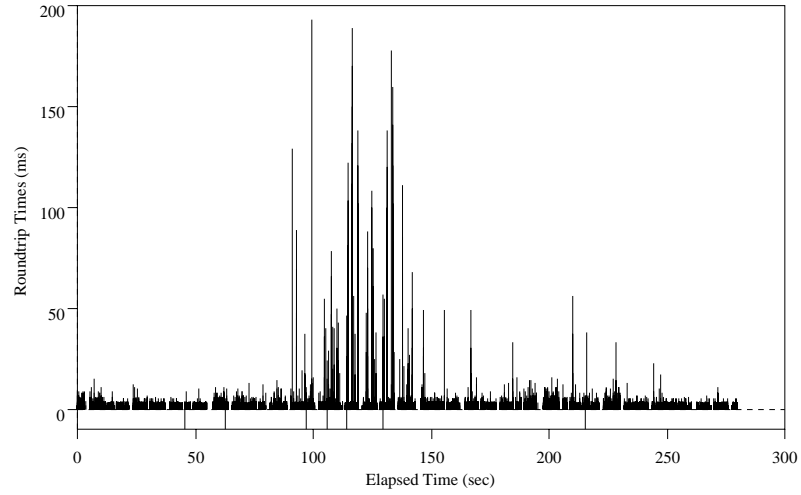


Figure 3.4: Roundtrip times over a 2 hop path within UVAnet.

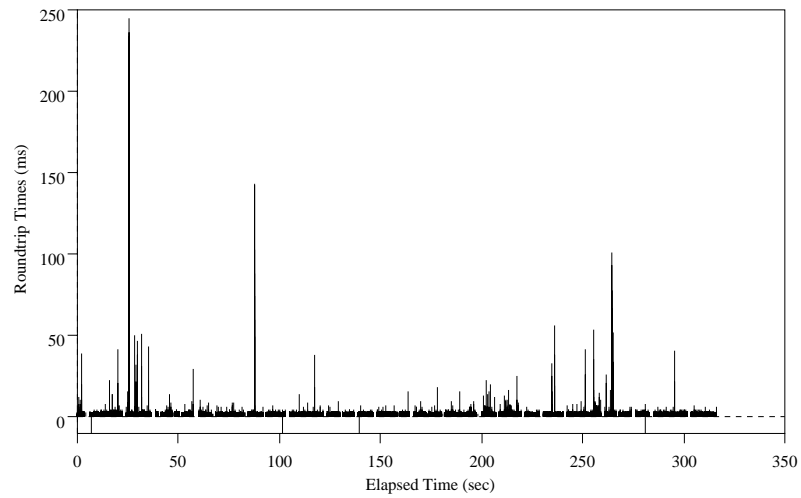


Figure 3.5: Roundtrip times over a 3 hop path within UVAnet.



### 3.3. Campus and Local Area Traffic Model 36

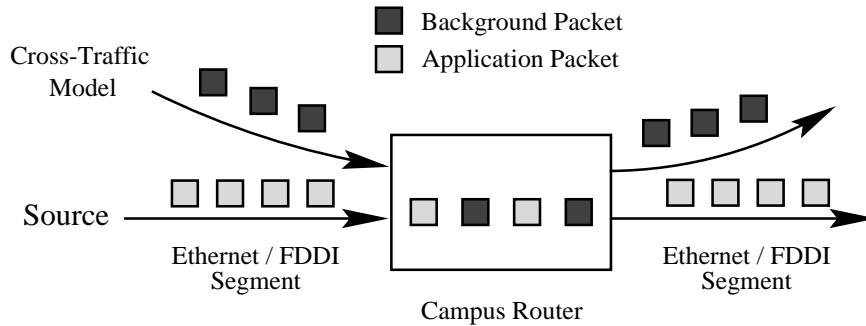


Figure 3.6: Campus background traffic approach.

afternoon. Table 3.1 shows that Ethernet drop rates are between 0% and 0.75%, and the FDDI interfaces experienced no drops. We observed that drops are not correlated with the interface's forwarding rate. For example, Ethernet interface 8 had the highest forwarding rate of  $148.3KB/sec$  and a drop rate of 0.001%. However, Ethernet interface 11 had the lowest forwarding rate of  $3.1KB/sec$ ; however, it experienced a drop rate of 0.62%. Finally, Ethernet interface 12 had a moderate forwarding rate of  $83.7KB/sec$ , with the highest drop rate of 0.75%. In the campus network simulation, the Ethernet interface drop rate is set at 0.4% (which represents moderate Ethernet interface drop rates) and the FDDI drop rate is set at 0%. In addition, an error probability of 0.1% is added at each interface to model discards due to malformed packets, processor overload, and router updates<sup>2</sup>.

Next we consider how packet drops are correlated in time. Figure 3.7 shows the drop rate as a function of time (sec) for the three highest byte forwarding interfaces at the Olsson router. The figures show that packets are dropped in isolated bursts throughout the series. Thus, for simplicity, packet drops are modeled independently and not correlated within the router interfaces.

SNMP does not give fine-granularity data on router input/output queuing, processing, or media access contention delays. Therefore, the router delay model is based on end-to-

---

<sup>2</sup>During routing table updates, routing on the interface is temporarily suspended thereby causing packet drops.

### 3.3. Campus and Local Area Traffic Model 37

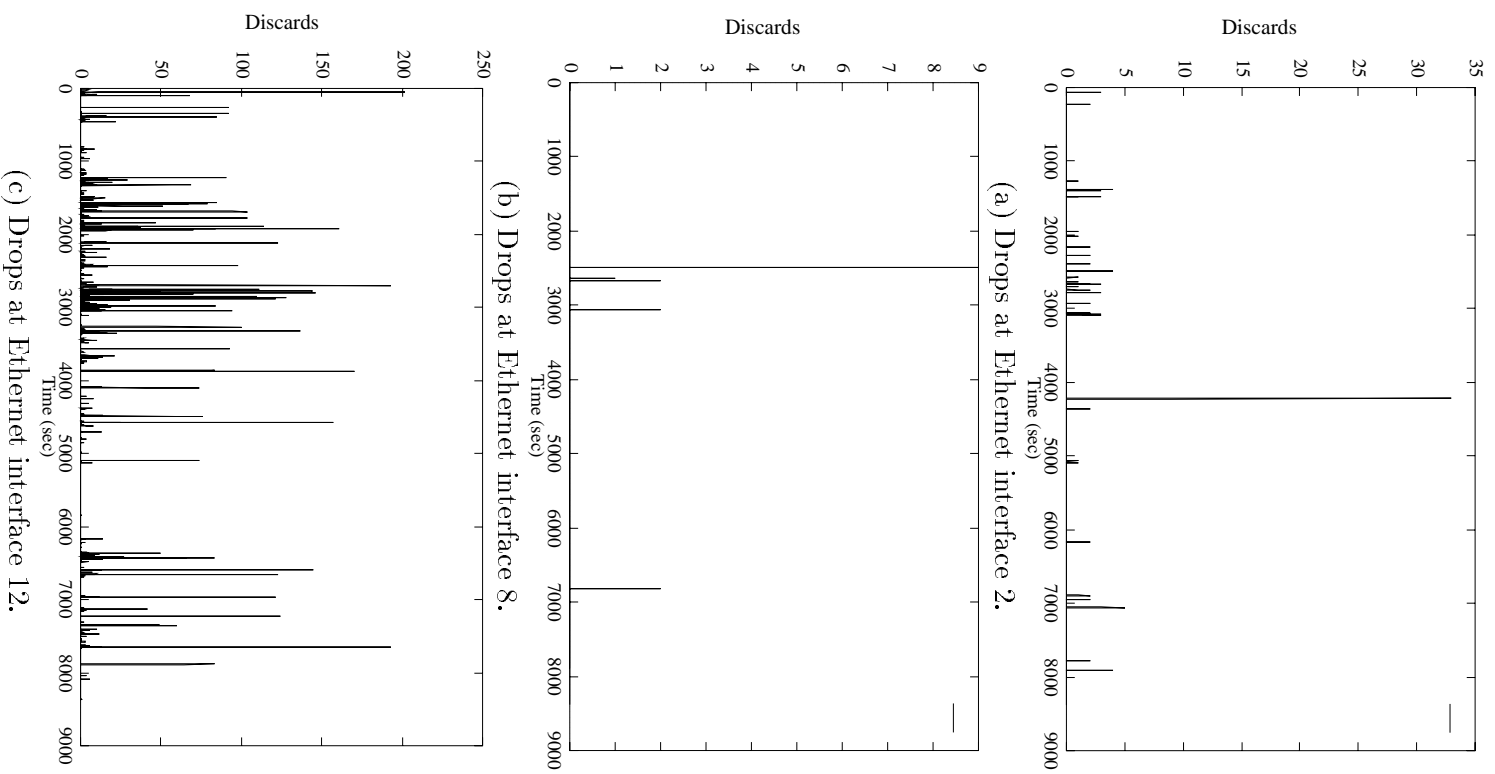


Figure 3.7: Packet drop series at Olsson Hall router Ethernet interfaces 2, 8, and 12.

### 3.3. Campus and Local Area Traffic Model 38

Interface	Packets/Sec	KBytes/Sec	Drop %
Ethernet Interface 1	15.4	11.7	0
Ethernet Interface 2	54.7	29.3	0.05
Ethernet Interface 3	9.0	6.3	0
Ethernet Interface 4	36.1	18.0	0.07
Ethernet Interface 5	16.6	6.1	0
Ethernet Interface 6	125.5	18.3	0
Ethernet Interface 7	43.1	12.0	0
Ethernet Interface 8	256.0	148.3	0.001
Ethernet Interface 9	56.8	19.0	0.003
Ethernet Interface 10	46.3	21.6	0.18
Ethernet Interface 11	3.8	3.1	0.62
Ethernet Interface 12	243.0	83.7	0.75
Ethernet Interface 13	18.4	5.4	0.03
Ethernet Interface 14	14.1	6.2	0.1
FDDI Interface 1	359.2	86.0	0
FDDI Interface 2	487.1	80.8	0

Table 3.1: Olsson router utilization and drop statistics.

end packet delays of the UVAnet traces presented in [38]. Figures 3.4 and 3.5 show that transmission delays over a two and three hop route are steady except for occasional large delays (up to 250 ms.) We ignore the outlying delay spikes and assign each packet a uniform delay between 1 – 3 ms at each output port. Like the loss model, delays are not correlated between arrivals. In the worst case, the delay model gives a maximum delay of 12 ms between two stations within the campus network, and a maximum of 6 ms delay to the WAN access point.

Packet delay and drop patterns at campus routers are modeled independently from adjacent routers. For example, if a burst of packets arrives at the router attached to Ethernet

segment 1 in Figure 3.3, the burst is not correlated with arrivals at router 2, router 3, or the backbone router. Modeling the delays and drops independently in a campus network is acceptable because most LAN traffic is highly localized (e.g., fileserver and remote login); thus, losses and delays are not correlated between routers as they would be in a WAN environment.

### 3.4 Wide-Area Background Traffic Model

In wide-area networks, the load at a particular router is a function of the load at adjacent routers. Thus, the cross-traffic model employed within the campus network is not appropriate for wide-area models since packet delays and drops are not correlated between adjacent network routers. Instead, the model developed in this section takes a different approach by introducing background traffic at each WAN NAP representing the traffic generated by a contemporary campus network. Each packet is then routed in the WAN to a destination NAP, thereby correlating network load at adjacent routers. The statistical properties of the background traffic and destination NAP address distributions control the delay and packet drop rates within the WAN.

One common WAN background traffic generation approach is to aggregate a number of individual sources (e.g., using `tcplib` [31]) until the desired background load in the network is achieved. Although the application-level aggregation technique is highly accurate, it is not efficient enough to scale to large backbone network simulations, especially those designed to evaluate complex communication patterns such as reliable multicast.

In this research, we model WAN traffic on a per-campus granularity using the technique summarized in Figure 3.8. First, a self-similar process  $\mathcal{M}$  models the aggregate packet arrival distribution and correlation structure generated by campus network. We call this sample path, representing arrivals per 100 ms, the *aggregate stream*.  $\mathcal{P}$  then divides the aggregate stream into individual *campus streams* — one for each destination campus network access point within the WAN. Finally, a short-term burst process  $\mathcal{S}$  takes the arrivals

### 3.5. WAN Statistical Characterization Experiment Setup 40

generated by  $\mathcal{P}$  and creates a sample path with a high resolution timing granularity, e.g., arrivals per millisecond.

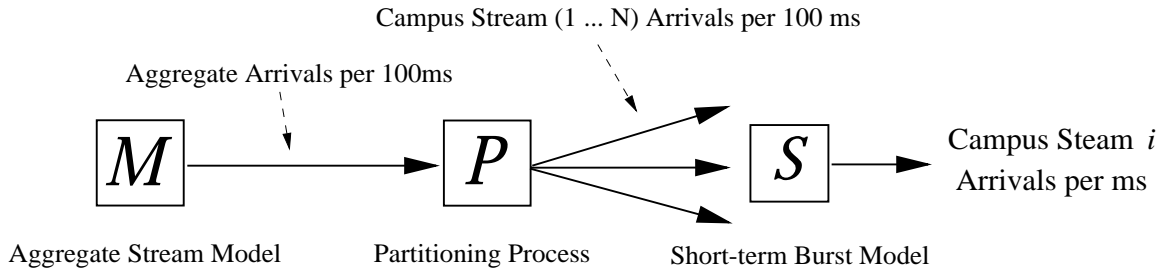


Figure 3.8:  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  traffic model.

Computational efficiency is a key advantage to the  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  model. In general, generating a large-scale background load that faithfully models the arrival density and correlation structure is difficult and computationally expensive. Our scheme requires only a single aggregate packet stream to be generated from each campus within the WAN. Further, each stage in the  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  model is efficient – the computational complexity of generating a sample path is  $O(N \cdot P \log P)$ , where  $N$  is the number of campus networks and  $P$  is the size of the sample path. The second key advantage is that the characteristics of the individual streams created by the partitioning process  $\mathcal{P}$  are a function of a small number of statistical properties of the aggregate stream. By expressing the background traffic in terms of the aggregate stream parameters, we can study the WAN under consideration for a range of congestion levels, independent of the underlying network topology and infrastructure.

### 3.5 WAN Statistical Characterization Experiment Setup

We use the statistical characteristics of the traffic generated by the University of Virginia campus network (UVAnet) to evaluate and motivate the design of the  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  traffic model. We chose UVAnet because it represents traffic generated by a large contemporary network, enterprise network, or small Internet service provider that would interconnect on

### 3.5. WAN Statistical Characterization Experiment Setup 41

a regional WAN. The data and analysis presented in this section develop the  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  background traffic model. However, the data also contributes to understanding the fractal behavior of wide-area network utilization as well as provide a benchmark to evaluate the accuracy of existing traffic models.

Our statistical analysis is based on 90-minute samples from a week-long trace of nearly one billion IP packets exchanged between UVAnet, Virginia Educational and Research Network (VERnet), and BBNplanet<sup>3</sup> (at the time, UVA's global Internet access provider). The network monitor used to collect the trace consists of a powerful workstation<sup>4</sup>, a customized kernel with large network buffers and background processes disabled, and a kernel-level packet filter [1]. The network monitor provides a timestamp resolution within 100  $\mu$ sec and an observed drop rate of less than 0.005% over the entire trace.

#### 3.5.1 Experiment Setup

Figure 3.9 shows the experimental setup which consists of three routers and a network monitor interconnected by an Ethernet hub. The VERnet and BBNplanet routers are each connected to three T1 links, while the UVAnet router is connected to UVA's backbone FDDI concentrator. The filter listens promiscuously on the Ethernet and capture all IP packets sent between the UVAnet, VERnet and BBNPlanet routers. The filter captures the IP header and saves the IP source, IP destination, timestamp, and size of each packet to disk. After compression, approximately six bytes are saved per packet.

Figure 3.10 depicts the nine-day packet trace captured by the packet filter. The figure plots the number of packets exchanged between the three networks per 100-second interval as a function of time. There are two periods where the monitor workstation went off-line. The first period occurred between 8PM Wednesday and 8AM Thursday due to a disk problem, and the second failure occurred at 11PM on the second Tuesday due to a campus-wide power outage. Two interesting observations about the data are: (1) the ratio of the

---

<sup>3</sup>SURAnet is the regional component of BBNplanet.

<sup>4</sup>Sun UltraSparc Model 170 with 192MB RAM and 8GB of disk space running Solaris 2.5.

### 3.5. WAN Statistical Characterization Experiment Setup 42

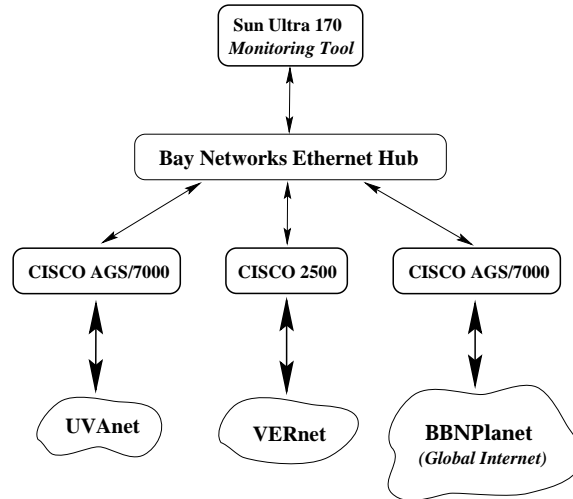


Figure 3.9: Experiment Setup

peak to the minimum data rate is approximately 8:1, which is bursty at this timescale, and (2) the packet rate is cyclical with periods of low utilization occurring around 5AM and peak utilization occurring around 4PM.

#### 3.5.2 Data Considered

The statistical analysis considers the packets leaving UVAnet destined for either BBNplanet or VERnet during the one-day period highlighted in Figure 3.10. The study is limited to outgoing packets because the end-goal of this section is to develop a model of packet arrivals generated at a wide-area backbone network access point (e.g., packet arrivals for a large campus network or small Internet service provider).

Figure 3.11 shows the number of packets generated by UVAnet per 10-second interval over the 27 hour trace. The network monitor experienced a single burst of drops during the 27 hour period when, just before 12PM, the monitor timed out for exactly ten seconds and dropped 9,784 packets. The statistical analysis focuses on the three 90-minute intervals highlighted in Figure 3.11. These intervals, namely the 2:15AM – 3:45AM (“2AM trace”),

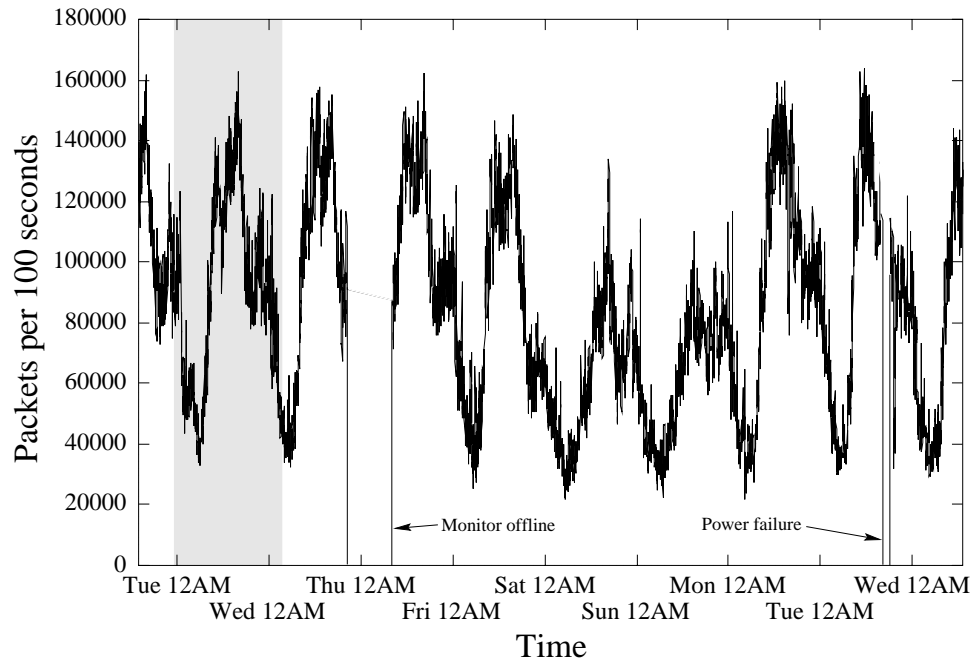


Figure 3.10: Packets per 100 seconds for 9 day packet trace.

2:00PM – 3:30PM (“3PM trace”), and 9:00PM – 10:30PM (“9PM trace”), correspond with periods of low, high and medium network utilizations, respectively, and because the arrival processes are stationary over the duration<sup>5</sup>. We present three traces from a single 27 hour trace. However, the statistical analysis results are consistent and representative of the data collected throughout the entire week.

### 3.6 Modeling the Packet Size

This section characterizes and presents a model of the density and correlation structure of the packet sizes generated by UVAnet. Figure 3.12 shows the empirical probability distribution of packet sizes for the 2AM, 3PM and 9PM traces. The density is presented on a logarithmic scale to highlight that a small number of packet sizes dominate the trace.

<sup>5</sup>Note that the packet arrival process must be stationary to evaluate its time-dependent properties.



### 3.6. Modeling the Packet Size 44

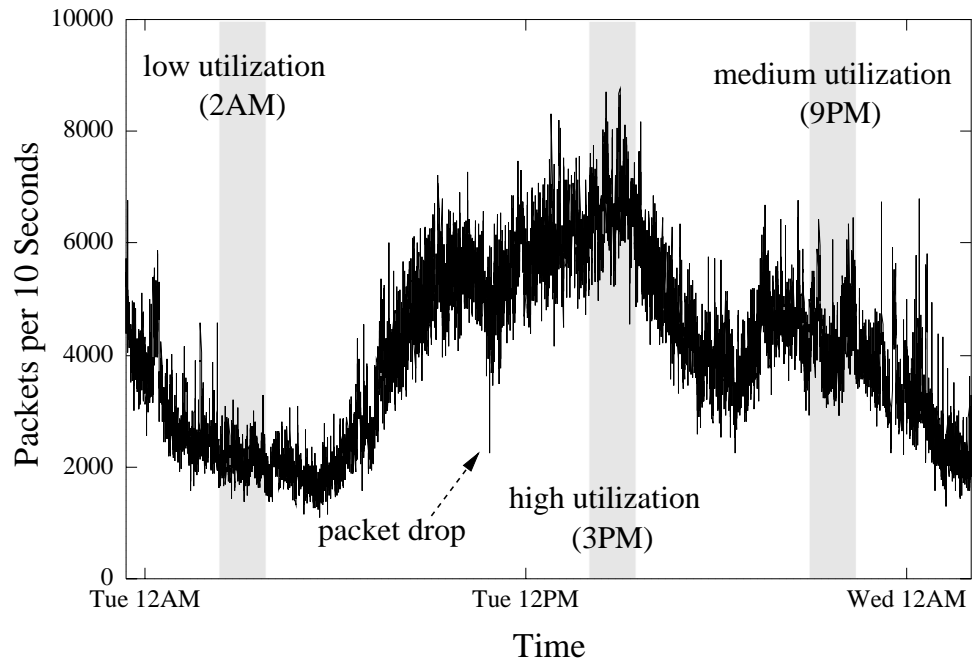


Figure 3.11: Packets per 10 second interval for first Tuesday packet trace.

In particular, approximately 75% of the packets are either 40 – 44 or 552 bytes in length. Inspection of the distribution also reveals “spikes” at 55, 60, 75, 144, 576, and 1500 byte packets, accounting for 12% of the packets. For all three traces, the average packet size was approximately 300 bytes. A key observation in Figure 3.12 is that the densities are nearly identical for all three traces, which shows that the distribution of packet sizes is independent of network utilization.

We next consider the correlation structure of the packet sizes. For a random process  $\{X_i\}_{i=0,1,\dots,N}$  with sample mean  $\bar{X}$  and sample variance of  $S^2$ , the autocorrelation function  $r$  can be estimated for all lag  $k$  as follows<sup>6</sup>:

---

<sup>6</sup>Autocorrelation describes the relationship of the size of packet  $i$  to packet  $i + k$ . If  $r(k) = 0$ , then the size of packet  $i$  has no impact on the size of packet  $i + k$ .

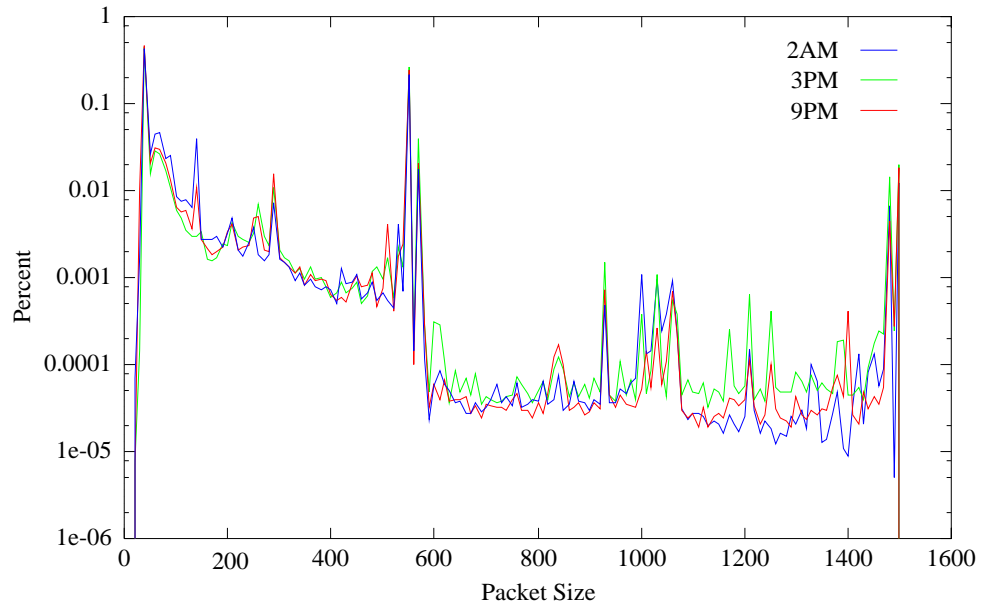


Figure 3.12: Probability density function of packet sizes.

$$r(k) = \frac{\sum_{i=1}^{N-k} (X_i - \bar{X})(X_{i+k} - \bar{X})}{(N-k)S^2} \quad (3.1)$$

Figure 3.13 gives the autocorrelation  $r(k)$  of the packet sizes plotted as a function of the lag  $k$  for each trace. Since the tail converges rapidly to 0, we can conclude that packet sizes are not correlated; i.e., the size of packet  $x_i$  has a negligible influence on the size of packet  $x_{i+1}, \dots, x_n$ . The lack of correlation can be explained by the nature of statistical multiplexing in IP networks. That is, packet sizes are most often highly correlated as they are generated by the application [81]. However, as the network statistically multiplexes a large number of independent connections, the correlation diminishes. For example, Figure 3.13 shows that the *9PM* trace has the most correlation, and the *3PM* trace has the least correlation. The correlation analysis shows that packet sizes can be faithfully modeled for a large campus network by independently choosing a packet size using the empirical density function shown in Figure 3.12.

### 3.7. Aggregate Wide-Area Network Traffic Model, $\mathcal{M}$ 46

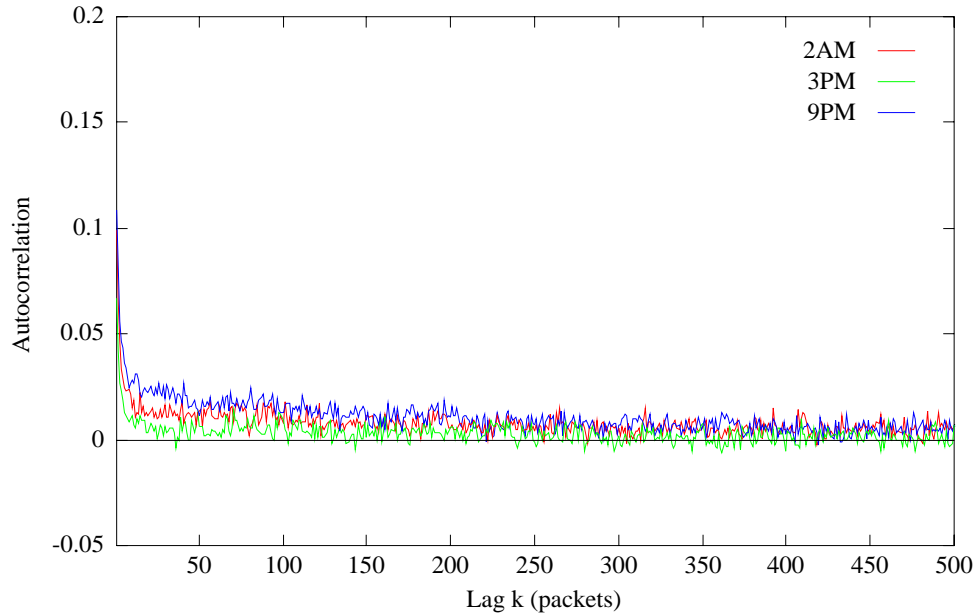


Figure 3.13: Autocorrelation function for packet sizes.

## 3.7 Aggregate Wide-Area Network Traffic Model, $\mathcal{M}$

This section develops a traffic model  $\mathcal{M}$  which generates a sample path such that the correlation structure and arrival distribution matches the UVAnet traces presented in Section 3.5. Accurately modeling the arrival correlation of the traffic stream generated by the campus network is critical since packet bursts dramatically affect the WAN packet drop rate, variation in network transmission delay, and available network throughput. In developing  $\mathcal{M}$ , we first use a self-similar traffic source to model the time-dependent component of the traces. Next, we transform the sample path to match the arrival density of the UVAnet traces.

### 3.7.1 Time-Dependent Statistical Properties

The time-dependent properties of the UVAnet streams are shown in Figure 3.14 by plotting the autocorrelation function of packet arrivals per 1 ms (note, the role of  $H$  is discussed next). In contrast to the packet size correlation, Figure 3.14 shows the correlation structure

### 3.7. Aggregate Wide-Area Network Traffic Model, $\mathcal{M}$ 47

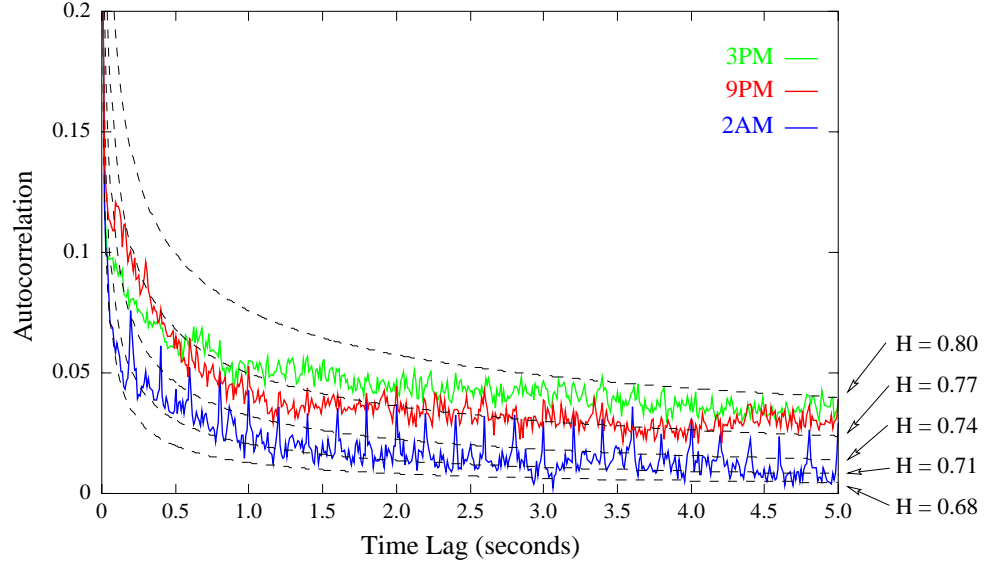


Figure 3.14: Autocorrelation function for the 2AM, 3PM and 9PM traces.

of packet arrivals is hyperbolically decaying, suggesting that the streams have long-range dependencies [29]. The important property of an arrival process with long-range dependencies is that the arrival burstiness is similar, independent of the time scale in which it is viewed (so-called *self-similar* processes). Formally, a stationary process  $\{X_i \mid i = 0, 1, \dots, \infty\}$  and its associated *aggregated arrival processes*  $\{X_i^{(m)} \mid m = 1, 2, \dots, \infty\}$  given by:

$$X_i^{(m)} = 1/m \sum_{k=im}^{i(m+1)-1} X_k \quad (3.2)$$

is *exactly second-order self-similar* if the autocorrelation  $r^{(m)}(k)$  of each aggregated process is given by [57]:

$$r^{(m)}(k) = r(k), k \geq 0 \quad (3.3)$$

and the variance is given by [57]:

$$\text{Var}(X^{(m)}) = \text{Var}(X)m^{-2(1-H)} \quad (3.4)$$

The degree of self-similarity is expressed by the Hurst parameter  $H$  in equation 3.4.  $H$  varies between 0.5 and 1, where a larger value indicates a higher degree of self-similarity. For a short-range dependent process, such as the Poisson-based models in [46, 51], the Hurst parameter will be approximately 0.5; thus, by (4), the correlation of a Poisson process will fall off as  $1/m$  where  $m$  is called the *aggregation level*. Using the reference curves in Figure 3.14 we see that the correlation structure of the traces correspond to self-similar processes with  $H$  between 0.70 and 0.80; thus, they can not be accurately modeled with a Poisson-based process. These results are consistent with studies showing the self-similarity of LAN traffic which have estimated the Hurst parameter as high as 0.82 [57].

### 3.7.2 Generating Self-Similar Traffic

The literature gives number of traffic models that generate long-range dependent traffic (see [45, 57, 80] and the references therein). In developing  $\mathcal{M}$ , we model a particular class of self-similar traffic called *Fractional Gaussian Noise* (FGN) [12] using the Fast Fourier Transform (FFT) method developed by Paxson [79]. The advantage to a FGN process is that the degree of self-similarity can be expressed solely by the Hurst parameter. We choose the FFT method (given in Appendix A.1) because an approximate FGN sample path of length  $P$  can be efficiently generated in  $O(P \log P)$ . Efficiency is crucial since a large simulation must generate potentially millions of packets per second; thus, exact self-similar processes based on fractional auto-regressive integrated moving averages (F-ARIMA) [7], alternating renewal processes [57] and M/G/ $\infty$  queues [29] are precluded because of their computational complexity.

### 3.7.3 Modelling the Arrival Density

The next step in  $\mathcal{M}$  is to transform the path such that the arrival density matches the UVAnet stream. Using a maximum likelihood estimator (MLE), the distribution of packet arrivals for UVAnet per 100 ms interval is fitted to the Pareto, Gamma, Weibull and Log-

### 3.7. Aggregate Wide-Area Network Traffic Model, $\mathcal{M}$ 49

normal distributions<sup>7</sup>. The goodness of fit is evaluated by plotting the quantiles of the empirical data against the quantiles (Q-Q) of the fitted distribution. We model the arrival density using a log-normal distribution because it faithfully captures the packet arrival process for the entire range of the distribution as shown in the Q-Q plots presented in Figure 3.15.

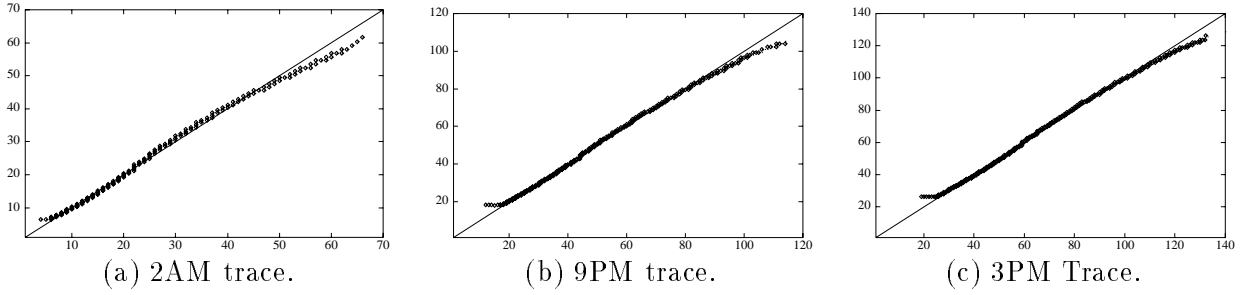


Figure 3.15: Q-Q plots of 2AM, 9PM and 3PM empirical traces versus fitted log-normal distributions.

The FFT algorithm produces a Normally distributed  $(0, 1)$  FGN sample path with long-range dependence corresponding to the Hurst parameter  $H$ . Next,  $\{x_0, \dots, x_{n-1}\}$  is transformed such that it is lognormally distributed with mean  $\overline{M}$  and variance  $\overline{V^2}$  as follows: First, construct  $\{x'_0, \dots, x'_{n-1}\}$  such that  $x'_i = x_i\sqrt{\sigma^2} + \mu$ ; where  $\mu$  and  $\sigma^2$  are given by:

$$\mu = \ln \overline{M} - \frac{1}{2} \ln \left\{ 1 + \frac{\overline{V^2}}{\overline{M}^2} \right\} \quad (3.5)$$

$$\sigma^2 = \ln \left\{ 1 + \frac{\overline{V^2}}{\overline{M}^2} \right\} \quad (3.6)$$

After this transformation,  $\{x'_0, \dots, x'_{n-1}\}$  is  $\text{Normal}(\mu, \sigma^2)$ . Next, construct  $\{x''_0, \dots, x''_{n-1}\}$  such that  $x''_i = \exp(x'_i)$ . The sample path  $\{x''_0, \dots, x''_{n-1}\}$  is now lognormal with mean  $\overline{M}$  and variance  $\overline{V^2}$ .

---

<sup>7</sup>These distributions are commonly chosen to model LRD processes because they have a heavy-tail [80].

### 3.7. Aggregate Wide-Area Network Traffic Model, $\mathcal{M}$ 50

#### 3.7.4 $\mathcal{M}$ Evaluation

Table 3.2 gives the Hurst, mean ( $\overline{M}$ ) and variance ( $\overline{V^2}$ ) parameters used by  $\mathcal{M}$  to model the UVAnet traces.  $\overline{M}$  and  $\overline{V^2}$  were obtained using a maximum likelihood estimator (MLE) for the arrival process measured in the UVAnet traces. We used the semi-parametric algorithm developed in [56] to approximate the Hurst parameter at  $H = 0.80$ .

Trace	Mean Packets/100 ms ( $\overline{M}$ )	Variance ( $\overline{V^2}$ )	Hurst ( $H$ )
2AM Trace	23.49	96.67	0.80
3PM Trace	60.73	272.39	0.80
9PM Trace	46.16	193.43	0.80

Table 3.2: Parameters used to model UVAnet arrivals.

In Figure 3.16, the distribution of packet arrivals per 100 ms for the three hour-long UVAnet traces are shown in solid lines, while the dashed lines give the arrival distribution as generated by  $\mathcal{M}$ . As the figure shows, the fit is excellent.

The time-dependent accuracy of the synthetic stream generated by  $\mathcal{M}$  is evaluated using log-variance plots. A log-variance plot gives the degree of burstiness of an arrival process over multiple time scales by plotting the logarithm (base 10) of the normalized variance of the aggregated arrival process  $X^{(m)}$  against the logarithm of its aggregation level,  $m$ . Figure 3.17 shows log-variance plots for the synthetic traces in green, and the empirical traces in red. The figure shows that the variance of the arrivals for all three traces decay slowly in proportion to a self-similar process with  $H = 0.65$  for small aggregation levels, and asymptotically as a self-similar process with  $H = 0.85$ . Further,  $\mathcal{M}$  faithfully reproduces the correlation structure of the UVAnet traces across the entire aggregation range.

### 3.7. Aggregate Wide-Area Network Traffic Model, $\mathcal{M}$ 51

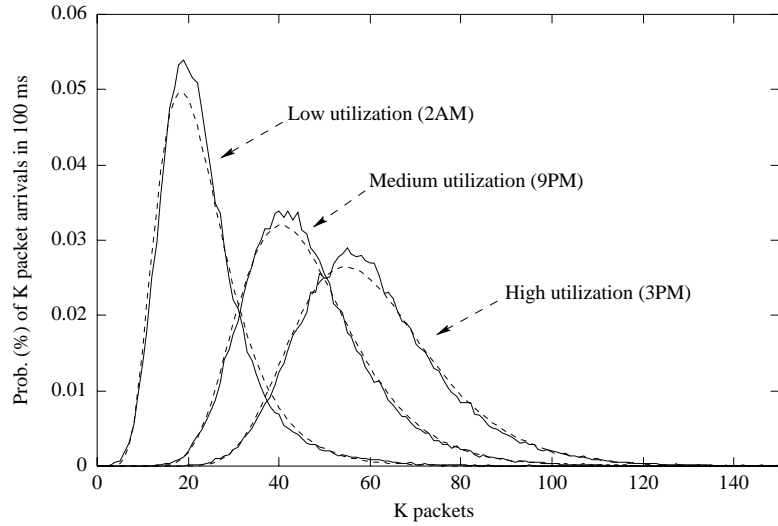


Figure 3.16: Arrival distribution of empirical traces (solid lines) and synthetic (dashed lines) for low, medium, and high network utilizations.

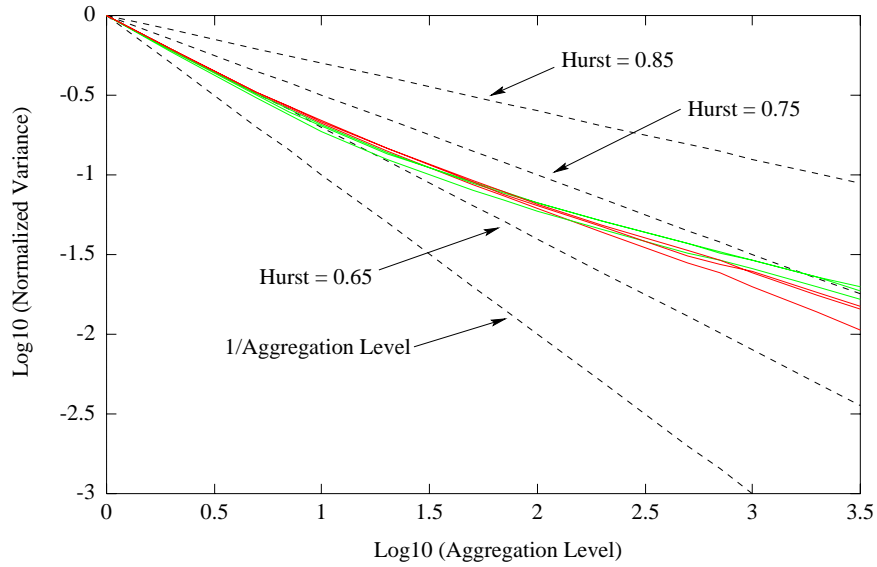


Figure 3.17: Log variance plot of synthetic (green lines) versus empirical (red lines) for 2AM, 9PM and 3PM UVA-net traces.



### 3.8 Partitioning Model $\mathcal{P}$ and Short-term Arrival Model $\mathcal{S}$

In the context of the  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  approach illustrated in Figure 3.8, the previous section developed a computationally efficient method  $\mathcal{M}$  to accurately model the traffic departing a campus network. This section develops the remaining two processes  $\mathcal{P}$  and  $\mathcal{S}$  which (1) partition the aggregate stream by assigning a destination network address to each arrival, and (2) distribute the substream packet generated by  $\mathcal{P}$  into arrivals per millisecond. We then evaluate the accuracy of the  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  approach by comparing the arrival density and correlation structure of synthetic traces to an example partitioning of the UVAnet traces.

#### 3.8.1 Partitioning Model $\mathcal{P}$

$\mathcal{P}$  takes as input a set of  $m$  target arrival distributions  $\{d_1, d_2, \dots, d_m\}$ , where  $m$  gives the number of campus destinations (NAPs) on the WAN. Each target distribution function  $d_i$  is defined as a set of probabilities  $\{p_{i,0}, p_{i,1}, \dots, p_{i,n_i}\}$ , where  $p_{i,k}$  is the probability of  $k$  packet arrivals on substream  $i$  during a time interval of length  $\tau$ . The goal of  $\mathcal{P}$  is to sample the path generated by  $\mathcal{M}$  into  $m$  substreams such that the packet arrival distribution for each substream  $i$  matches the target arrival density  $d_i$ .

The aggregate traffic stream generated by  $\mathcal{M}$  is expressed as a sequence of packet arrivals  $\{x_0, x_1, \dots, x_{n-1}\}$ , where  $x_j$  gives the number of packet arrivals during time interval  $[\tau j, (\tau + 1)j]$  (note  $\tau = 100$  ms.)  $\mathcal{P}$  constructs a set of  $m$  sequences  $\{Y_1, Y_2, \dots, Y_m\}$ , where  $Y_i = \{y_{i,0}, y_{i,1}, \dots, y_{i,n-1}\}$ , such that  $x_j = \sum_{i=1}^m y_{i,j}$  as follows: For each target density function  $d_i$ , a discrete-time birth-death process (i.e., Markov chain) is constructed with  $N_i$  states such that the steady-state probability that process  $i$  is in state  $k$  is exactly  $p_{i,k}$ . The “birth” and “death” transition probabilities  $\lambda$  and  $\mu$  are determined using the well-known relationship:

$$p_{i,k} = p_{i,0} \prod_{l=1}^k \frac{\lambda_{i,l-1}}{\mu_{i,l}} \quad (3.7)$$

### 3.8. Partitioning Model $\mathcal{P}$ and Short-term Arrival Model $\mathcal{S}$ 53

where  $\lambda_{i,l}$  is the probability of moving from state  $l$  to state  $l + 1$ , and  $\mu_{i,l}$  is the probability of moving from state  $l$  to state  $l - 1$ .

For each time period  $j$ , the Markov chains are independently modulated to obtain  $m$  state sequences  $\{s_{i,j} \mid 1 \leq i \leq m, 0 \leq j \leq n - 1\}$ , where  $s_{i,j}$  denotes the state of the  $i^{\text{th}}$  chain during the  $j^{\text{th}}$  time interval. The Markov chains are normalized such that  $\sum_{i=1}^m p_{s_{i,j}} = 1$  for all  $j$ . Finally, each packet from the aggregate stream is independently and randomly assigned a campus network address, where the probability that a given packet is assigned  $NAP_i$  is given by  $\frac{s_{i,j}}{\sum_k s_{k,j}}$ .

Note that model  $\mathcal{P}$  satisfies the requirement for efficiency since (1) each Markov chain is modulated independently from the others, and (2) the cost of partitioning an aggregate stream arrival generated by  $\mathcal{M}$  involves rotating the Markov chains and generating a random number to determine the destination campus network address ( $NAP_i$ ) for each packet in  $x_j$ .

#### 3.8.2 $\mathcal{S}$ : Short-term Packet Arrival Model

Next we develop a model  $\mathcal{S}$  which distributes the packet arrivals generated by  $\mathcal{P}$  into short term arrivals (i.e., arrivals per ms). One approach is to uniformly distribute packet arrivals across the 100 ms interval, or use Poisson-based interarrivals. However, the analysis in [80] shows that arrivals, when viewed at millisecond granularity, occur in bursts rather than a continuous flow.

$\mathcal{S}$  models the short-term arrival using the well-known packet-train model developed by Jain [51]. The packet-train model considers packet arrivals as a sequence of bursts (so-called *trains*), as opposed to independent events (so-called *cars*). A train is defined as a series of arrivals, such that the elapsed time between any two packets does not exceed the maximum allowable intercar gap (MAIG). The *train size* is defined as the number of packets within the train, and the *train length* is defined as the elapsed time from the first to last packet. As an example, Figure 3.18 illustrates the packet-train concept for the 2AM, 3PM, and 9PM

### 3.9. Application of the $(\mathcal{M}, \mathcal{P}, \mathcal{S})$ Model 54

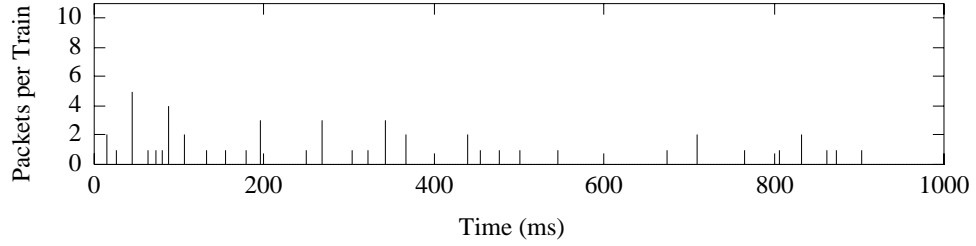
campus substreams with addresses between 192.0.0.0 - 200.0.0.0. The figures plot the train size using a 5 ms MAIG for an arbitrary one-second interval.

The short term model  $\mathcal{S}$  uses the packet-train approach to transforming the arrival path  $X_{i,j}$  generated by  $\mathcal{P}$  (where  $i$  refers to  $i^{\text{th}}$  substream, and  $j$  refers to the  $j^{\text{th}}$  sample) into arrivals per *ms*. Since the average train size is related to the mean of the substream, the average number of packets per train must be determined for each substream.  $\mathcal{S}$  computes the mean of each *substream $_i$*   $\{\chi_1, \chi_2, \dots, \chi_m\}$  using the target arrival distributions parameters  $\{d_1, d_2, \dots, d_m\}$  from  $\mathcal{P}$ . The average number of packets per train is shown in Figure 3.19 as a function of the mean of the campus substreams for the 2AM, 3PM and 9PM traces. Using the equation  $l(x) = 12.5x + 1.35$  shown in Figure 3.19, the average packets per *train $_i$*  is given by  $l(\chi_i)$ . For each arrival  $x_{i,j}$ , the packets are distributed throughout the 100 ms interval by constructing a set of trains  $t_1, t_2, \dots, t_n$  such that the arrival time of  $t_n$  is  $\leq 100$  ms from the arrival time of  $t_0$ . The size of packet train  $t_k$  is determined using a Poisson distribution with mean  $l(\chi_i)$ . The train interarrival time (i.e., the elapsed time between train  $r_k$  and  $r_{k+1}$ ) is determined by a Poisson distribution with mean  $\frac{x_{i,j}}{l(\chi_i)}$ .

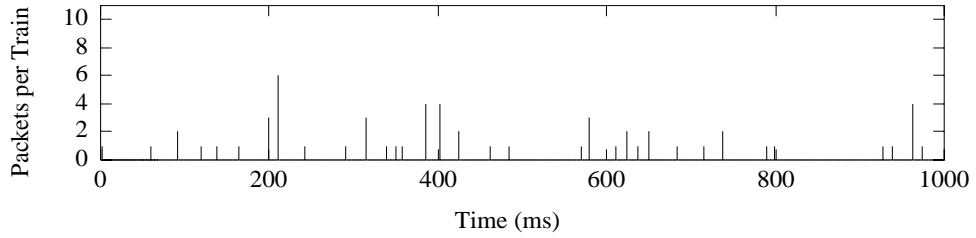
### 3.9 Application of the $(\mathcal{M}, \mathcal{P}, \mathcal{S})$ Model

This section applies the  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  traffic model to an example partitioning of the UVAnet traces described in Section 3.5. Table 3.3 gives the network masks used to partition the aggregate stream into fourteen campus substreams based on destination IP address (for a review of IP addressing, refer to [28]). The table gives the percentage of packets that each substream contributes to the aggregate stream for the 2AM, 3PM and 9PM traces. The Class A and Class D/E address spaces are each assigned their own campus stream, while the Class B and C address space is partitioned along bits 2-5 into twelve streams of equal size with respect to the number of network addresses they cover. Although this partitioning is arbitrary, it is sufficient to illustrate the  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  model and compare the correlation and arrival density of the synthetic streams generated by  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  to the UVAnet traces.

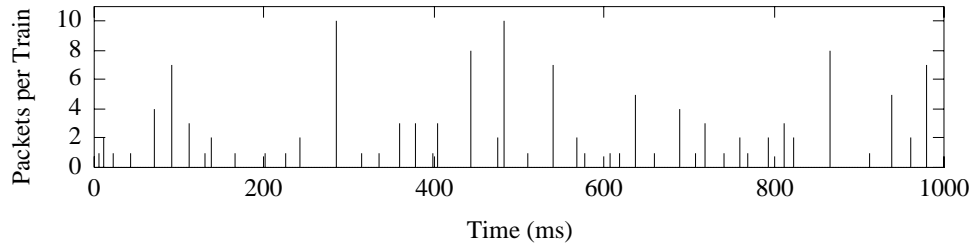
### 3.9. Application of the $(\mathcal{M}, \mathcal{P}, \mathcal{S})$ Model 55



(a) 2AM Trace - Low Utilization.



(b) 9PM Trace - Medium Utilization.



(c) 3PM Trace - High Utilization.

Figure 3.18: Packet-train time series for (a) 2AM, (b) 9PM and (c) 3PM campus substreams with addresses 192.0.0.0 - 199.255.255.255.

### 3.9. Application of the $(\mathcal{M}, \mathcal{P}, \mathcal{S})$ Model 56

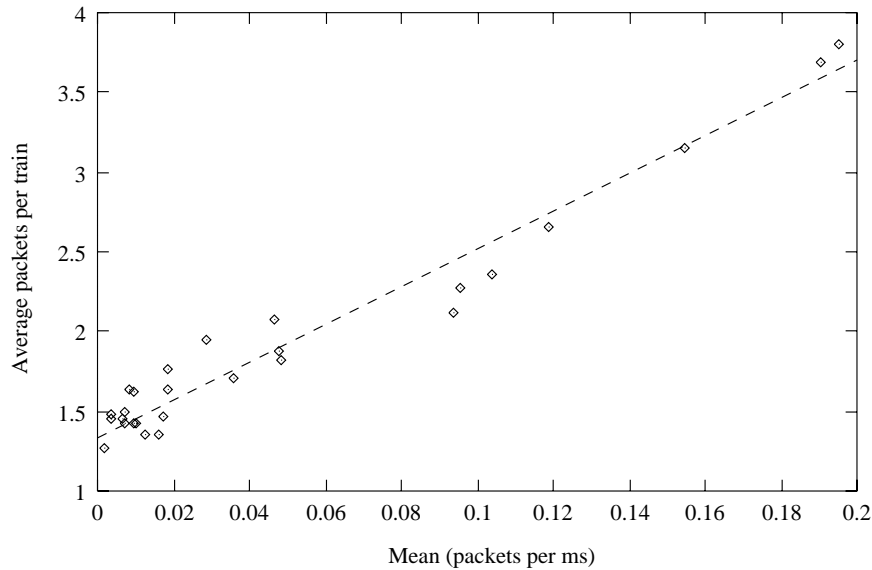


Figure 3.19: Number of packets in a train plotted against the mean of the campus substream (in packets per 1ms).

The arrival distributions  $D = \{d_{class\ a}, d_{128-135}, \dots, d_{class\ d}\}$ , and the packet train sizes  $S = \{s_{class\ a}, s_{128-135}, \dots, s_{class\ d}\}$  are constructed for the UVAnet 3PM campus substreams created by the partitioning given in Table 3.3. Next, we construct synthetic streams for the 3PM trace using  $\mathcal{M}(\overline{M} = 60.73, \overline{V^2} = 272.39, H = 0.80)$ <sup>8</sup>,  $\mathcal{P}(D)$ , and  $\mathcal{S}(S)$ .

Figure 3.20 compares the arrival density of the synthetic stream (dashed lines) with the empirical streams (solid lines) for the five largest streams in the 3PM trace<sup>9</sup>. For the three largest streams, the arrival density of the synthetic stream matches the empirical stream closely. For the smaller streams, the fit is also good except the synthetic traffic slightly underestimates the peak and overestimates the tail of the distribution. For comparison, Figure 3.21 shows a simple, first-order partitioning approach which divides the aggregate

<sup>8</sup>These parameters were derived in Section 3.7.

<sup>9</sup>For clarity of presentation, the remaining smaller streams are omitted.

### 3.9. Application of the $(\mathcal{M}, \mathcal{P}, \mathcal{S})$ Model 57

Filter Mask	2AM Trace	9PM Trace	3PM Trace
0.0.0.0 – 127.255.255.255 (Class A)	1.6%	1.6%	1.7%
128.0.0.0 – 135.255.255.255 (Class B)	20%	20%	21%
136.0.0.0 – 143.255.255.255 (Class B)	6.9%	5.9%	3.9%
144.0.0.0 – 151.255.255.255 (Class B)	3.0%	3.0%	2.4%
152.0.0.0 – 159.255.255.255 (Class B)	4.2%	7.7%	6.3%
160.0.0.0 – 167.255.255.255 (Class B)	3.0%	3.0%	2.4%
168.0.0.0 – 175.255.255.255 (Class B)	0.6%	1.4%	1.0%
176.0.0.0 – 183.255.255.255 (Class B)	0.0%	0.0%	0.0%
184.0.0.0 – 191.255.255.255 (Class B)	0.0%	0.0%	0.0%
192.0.0.0 – 199.255.255.255 (Class C)	21%	26%	22%
200.0.0.0 – 207.255.255.255 (Class C)	40%	32%	39%
208.0.0.0 – 215.255.255.255 (Class C)	0.0%	0.1%	0.2%
216.0.0.0 – 223.255.255.255 (Class C)	0.0%	0.0%	0.0%
224.0.0.0 – 255.255.255.255 (Class D/E)	0.3%	0.1%	0.2%

Table 3.3: Network filter mask and percent of traffic for 2AM, 9PM and 3PM traces.

stream based on the relative probabilities given in Table 3.3. Note that the first-order approach does not accurately model the arrival density for any of the 3PM campus streams.

Figure 3.22 shows the log-variance plot for the empirical and synthetic substreams for the 3PM traces. As the figure shows, the self-similarity of the synthetic streams in Figure 3.22(b) compares favorably to the empirical log-variance plots depicted in Figure 3.22(a) for all aggregation levels. Note that in the synthetic traffic, the initial slope of the variance falls off as  $1/m$ , underestimating the empirical data. This effect is created by the short-term packet model  $\mathcal{S}$  which uses a Poisson distribution to model the short-term packet arrivals. However, at one order of magnitude (i.e., arrivals per 100 ms) on the log-variance plot, the slope begins to fall off as a self-similar process with a Hurst parameter of 0.80. The increase is driven by  $\mathcal{M}$ , which generates self-similar packet arrivals at the 100 ms granularity. Finally, at three orders of aggregation, the variance of the synthetic begins to underestimate the

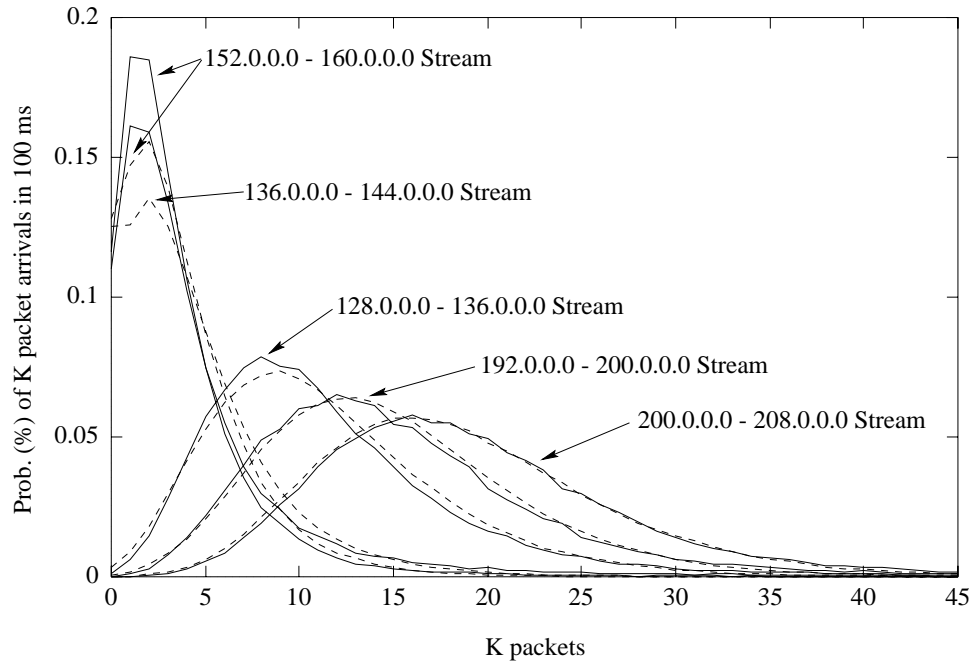


Figure 3.20: Probability density function for empirical campus streams (solid lines) and model  $P$  (dashed lines) for 3PM trace.

long-term burstiness. This decrease is an effect of the Markov chain (used in model  $\mathcal{P}$ ) smoothing the long-range burstiness of the traffic generated by  $\mathcal{M}$ . Overall, however, the synthetic streams generated by  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  closely match both the arrival distribution and correlation structure of the empirical traces.

### 3.10 SURAnet and vBNSnet Simulation Parameters

This section integrates the  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  background traffic model into the SURAnet and vBNSnet network models presented in Sections 3.1-3.3. First, Section 3.10.1 derives traffic model parameters for both SURAnet and vBNSnet, discusses implementation issues, and gives traffic model customizations. Next, Section 3.10.2 presents the set of parameters used to drive SURAnet and vBNSnet network load from light to peak utilization. Finally, Sec-

### 3.10. SURAnet and vBNSnet Simulation Parameters 59

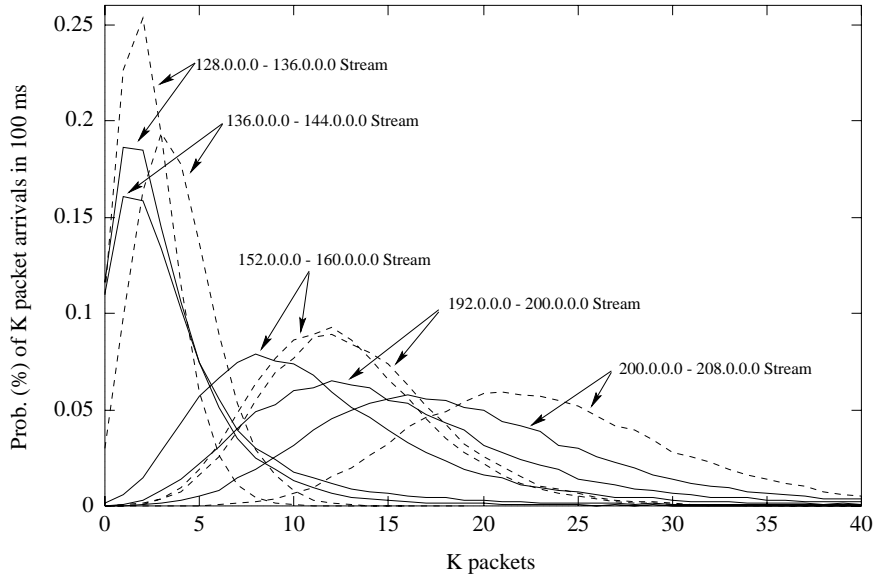


Figure 3.21: Simple partitioning approach for 3PM trace using empirical probability.

tion 3.10.2 presents the observed end-to-end network delays and drops in the SURAnet and vBNSnet simulations for the range of background load considered.

#### 3.10.1 SURAnet and vBNSnet $(\mathcal{M}, \mathcal{P}, \mathcal{S})$ Model Parameters

Incorporating the  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  traffic model into the SURAnet and vBNSnet network environments involves deriving the model parameters for each submodel  $\mathcal{M}$ ,  $\mathcal{P}$ , and  $\mathcal{S}$ . Each submodel is discussed below:

- *Model  $\mathcal{M}$* : As presented in Section 3.7,  $\mathcal{M}$  takes three input parameters: mean  $\overline{M}$  (in packets per 100 ms), variance  $\overline{V^2}$ , and Hurst parameter  $H$ .  $\overline{M}$  and  $\overline{V^2}$  characterize the arrival density generated by a campus network, whereas  $H$  is used to control the self-similarity of the traffic. Section 3.7 estimated  $\overline{M}$  and  $\overline{V^2}$  for UVAnet using a lognormal MLE. The data show that for UVAnet,  $\overline{V^2}$  can be approximated as 4.2 times  $\overline{M}$ ; thus, in both the SURAnet and vBNSnet simulations  $\overline{V^2} = 4.2 \times \overline{M}$ . Section 3.7 showed



### 3.10. SURAnet and vBNSnet Simulation Parameters 60

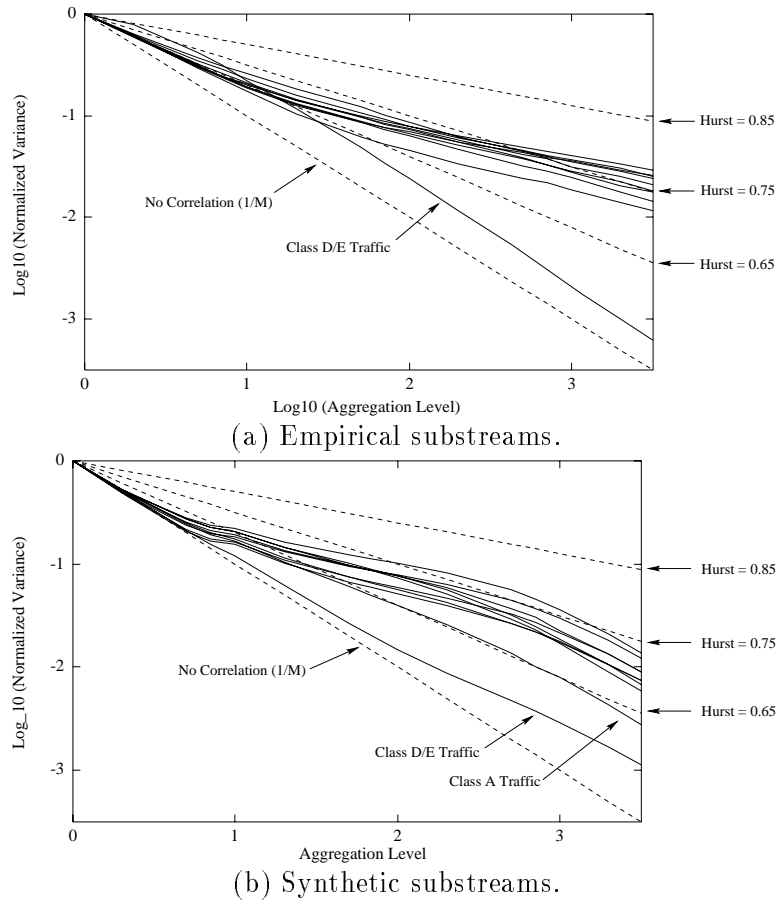


Figure 3.22: Log-variance plot of 3PM component substreams. Note the correlation structure of the synthetic (b) match the empirical (a).

that  $H = 0.80$  faithfully models the self-similarity of UVAnet over the entire range of network utilization. Thus,  $H$  is fixed at 0.80 for all values of  $\overline{M}$  in the SURAnet and vBNSnet simulations, and the background load is characterized solely by  $\overline{M}$ .

- *Model  $\mathcal{P}$* : As presented in Section 3.8.1,  $\mathcal{P}$  takes as input a set of probability density functions  $\{pdf_0, pdf_1, \dots, pdf_n\}$ ; where  $pdf_i$  represents the target arrival density for the substream destined for campus network access point  $i$ . Using  $\{pdf_0, pdf_1, \dots, pdf_n\}$ ,

### 3.10. SURAnet and vBNSnet Simulation Parameters 61

$\mathcal{P}$  partitions the sample path generated by  $\mathcal{M}$  into  $n - 1$  substreams. Section 3.8.1 derived  $\{pdf_{classA}, pdf_{128-135}, \dots, pdf_{224-255}\}$  by considering the substreams obtained by partitioning the UVAnet traces along class A, B, C, and D/E IP addresses. The PDFs used in the SURAnet and vBNSnet simulations are created in the same manner, except the partitioning tables are customized for each network model. Tables 3.4 and 3.5 are used to derive  $PDF_{SURAnet}$  and  $PDF_{vBNSnet}$  respectively. Each table is discussed below:

The SURAnet partitioning table is constructed such that 10% of network traffic is local to SURAnet, and the remaining 90% is routed to the global Internet. Each component substream is defined by first considering (where possible) the IP address blocks assigned to SURAnet campus networks. For example, address block 130.207.0.0 is assigned to the GIT campus; thus, it is included in the GIT substream definition 130.107.0.0 – 130.255.255.255. Next, the capacity of infrastructure connecting the campus network to SURAnet is considered and assigned an address block with proportional traffic volume. In addition to the 17 campus networks, two additional nodes UMD-NXP and GIT-NXP are added as the SURAnet network exchange points (NXPs) located at GIT and UMD. The  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  model is not employed at the UMD-NXP and GIT-NXP, since such traffic does not represent the traffic generated by a campus network. Instead, the NXPs simply reflect incoming packets to the SURAnet source. Finally,  $PDF_{SURAnet} : \{pdf_{UMD}, pdf_{NOF}, \dots, pdf_{UMD-NXP}\}$  is constructed by applying Table 3.4 to the UVAnet traces.

The vBNSnet partitioning is shown in Table 3.5. Since vBNSnet packet traces are not available, Table 3.5 is based on statistics reported in the April, 1997, vBNS management and operations report [6].  $PDF_{vBNSnet} : \{pdf_{PSC}, pdf_{NCAR}, \dots, pdf_{MAE}\}$  is constructed by applying Table 3.5 to the UVAnet traces.

- *Model S*: As described in Section 3.8.2, model  $\mathcal{S}$  distributes the arrivals per 100 ms generated by  $\mathcal{P}$  into arrivals per ms. In the SURAnet environment,  $\mathcal{S}$  is implemented

### 3.10. SURAnet and vBNSnet Simulation Parameters 62

SURAnet Node	# T-1 Links	Assigned Address Range	% Traffic
UMD	6	128.0.0.0 - 128.31.255.255	0.80%
NOF	3	128.32.0.0 - 128.83.255.255	0.71%
CTV	5	128.84.0.0 - 128.111.255.255	0.74%
WVU	4	128.112.0.0 - 128.123.255.255	0.51%
LEX	4	128.124.0.0 - 128.150.255.255	0.47%
GNU	4	128.151.0.0 - 128.168.255.255	0.52%
TAU	5	128.169.0.0 - 128.173.255.255	0.72%
MEM	2	128.174.0.0 - 128.182.255.255	0.27%
BIR	4	128.183.0.0 - 128.196.255.255	0.58%
KNX	3	128.197.0.0 - 128.217.255.255	0.41%
JCK	4	128.247.0.0 - 128.234.255.255	0.55%
NEO	2	128.248.0.0 - 128.255.255.255	0.32%
JKV	4	130.0.0.0 - 130.15.255.255	0.46%
MBJ	2	130.16.0.0 - 130.70.255.255	0.26%
AUB	2	130.71.0.0 - 130.90.255.255	0.21%
MSB	2	130.91.0.0 - 130.106.255.255	0.29%
GIT	14	130.107.0.0 - 130.255.255.255	1.55%
GIT-NXP	NA	0.0.0.0 - 127.0.0.0, 129.0.0.0-129.255.255.255, 131.0.0.0-191.255.255.255, 224.0.0.0-255.255.255.255	31.2%
UMD-NXP	NA	192.0.0.0 - 223.255.255.255	59.4%

Table 3.4: Network address ranges used to define SURAnet substreams.

exactly as described in Section 3.8.2. However, in the vBNSnet simulation,  $\mathcal{S}$  is modified such that if the average train interarrival time is less than 10 ms for the  $i^{th}$  arrival generated by  $\mathcal{P}$ , then  $\mathcal{S}$  distributes the packet arrivals using a Poisson distribution. The Poisson model is used because in the vBNSnet traffic model, packet arrival bursts can exceed 30K packets per second. Such an arrival process is not well suited for the packet train model which organizes the arrivals into bursts (i.e., trains)

### 3.10. SURAnet and vBNSnet Simulation Parameters 63

vBNSnet Node	Assigned Address Range	% Traffic
PSC	0.0.0.0 - 128.255.255.255 224.0.0.0 - 255.255.255.255,	8.23%
NCAR	129.0.0.0 - 131.255.255.255	9.46%
Chicago	132.0.0.0 - 137.255.255.255	9.35%
Denver	138.0.0.0 - 151.255.255.255	5.55%
Cornell	152.0.0.0 - 191.255.255.255	7.96%
SFC	192.0.0.0 - 192.255.255.255	11.0%
SDSC	193.0.0.0 - 198.255.255.255	13.81%
Cleveland	199.0.0.0 - 203.255.255.255	9.21%
Houston	204.0.0.0 - 204.255.255.255	8.63%
NCSA	205.0.0.0 - 205.255.255.255	6.94%
MAE	206.0.0.0 - 223.255.255.255	9.76%

Table 3.5: Network address ranges used to define vBNSnet substreams.

followed by periods of silence. Finally, in the vBNSnet environment, background packets are introduced into the network at a rate of  $1/arrival_j \mu\text{sec}$ , where  $arrival_j$  is the number of packets in the  $j^{\text{th}}$  generated by  $\mathcal{S}$  (note:  $0 \leq j < 100$ .) This step is necessary because a sub-millisecond timing granularity is required for the high-speed vBNSnet infrastructure.

#### 3.10.2 SURAnet and vBNSnet Delay and Drop Rates

Section 3.10.1 presented  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  model parameters for SURAnet and vBNSnet. The parameters are fixed for each simulation environment, except for  $\overline{M}$ , which drives the background traffic generated at each campus network. This section presents the range for  $\overline{M}$  used in the SURAnet and vBNSnet simulations.  $\overline{M}$  is parameterized such that congestion levels vary from light to peak congestion with network drop rates approaching 20% on the worst case network path. The 20% drop rate is chosen to provide a broad spectrum to

### 3.10. SURAnet and vBNSnet Simulation Parameters 64

evaluate protocol performance, and because it represents error rates found in congested Internet paths today.

#### 3.10.2.1 SURAnet Performance

The statistical characterization of wide-area packet arrivals presented in Sections 3.4-3.6 show that UVAnet generates between 20 and 60 packets per 100 ms. Thus,  $\overline{M} = 20$  at the UVAnet campus would generate traffic corresponding to light load, whereas  $\overline{M} = 60$  corresponds to heavy load. Since each campus network on SURAnet has various access speeds,  $\overline{M}$  scaled relative to the CTV NAP based on the access speed and traffic distribution given in Table 3.4. For example, MSB has 3 mbps access to SURAnet, whereas UVAnet has 5.5 mbps access. Thus, the proportion of traffic generated at CTV (given by  $\overline{M}_{CTV}$ ) nearly twice that of MSB ( $\overline{M}_{MSB}$ ).

In the SURAnet simulation environment, network load  $\overline{M}$  at the CTV NAP (also referred to as the UVA NAP) is considered between 10–70 packets per 100 ms. Although this range is slightly larger as observed in the UVAnet traces, it realizes the worst-case target network drop rate of approximately 20%.

Tables B.1 and B.2 in appendix B give the round trip times observed within the SURAnet simulation. The delay statistics were collected by having stations at each campus send a 100 byte probe message every 100 ms to a station located at each remote campus network. The remote station immediately reflected the probe message. As Tables B.1 and B.2 show, the delays under light load roughly correspond to the propagation delay in the WAN plus the campus network access delay (which averages 4 ms). For example, the delay between UVA and UMD averages 21 ms. Of the 21 ms, 16 ms is incurred within the campus networks, 2 ms is the link propagation delay within SURAnet, and 3 ms is SURAnet queuing/transmission delay. Under heavy load ( $\overline{M} = 70$ ), the average round-trip time is 203 ms with a peak delay of 970 ms between UMD and MSB. Under heavy load, delay is dominated by queuing delays within SURAnet. The SURAnet delays are shown graphically between UVA and selected

### 3.10. SURAnet and vBNSnet Simulation Parameters 65

campus networks in Figure 3.23. Note that the delays are roughly constant until  $\bar{M} = 30$ , then rise sharply until  $\bar{M} = 60$  at which point the marginal delay decreases because the network has reached saturation.

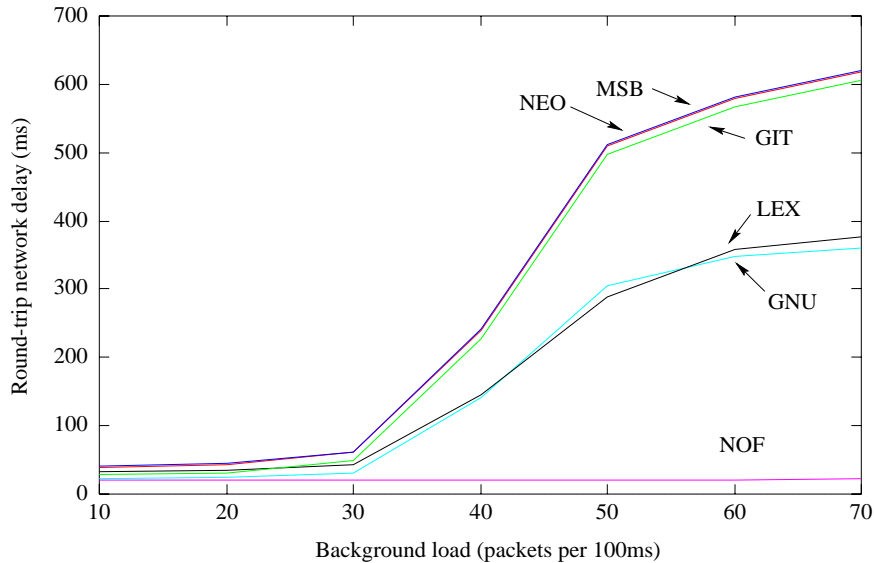


Figure 3.23: SURAnet round-trip delays for  $\bar{M} = 10 - 70$  background packets per 100 ms.

Tables B.3 and B.4 in appendix B give the network drop rates observed within the SURAnet simulation for light and peak load. Note that during light load ( $\bar{M} = 10$ ), roundtrip network drop rates are less than 8% on the worst-case path. However, at peak load ( $\bar{M} = 70$ ), round-trip drop rates are as high as 43%. Figure 3.24 plots the network drop rates from the CTV campus to selected sites. Note that the network drop rates remain fairly constant until  $\bar{M} = 40$ , then increase sharply. This is because utilization at the routers at GNU and GIT begin to approach capacity. Thus, the network performance between nodes on either side of GNU and GIT remains good; however, communication performance across these boundaries is relatively poor.

### 3.10. SURAnet and vBNSnet Simulation Parameters 66

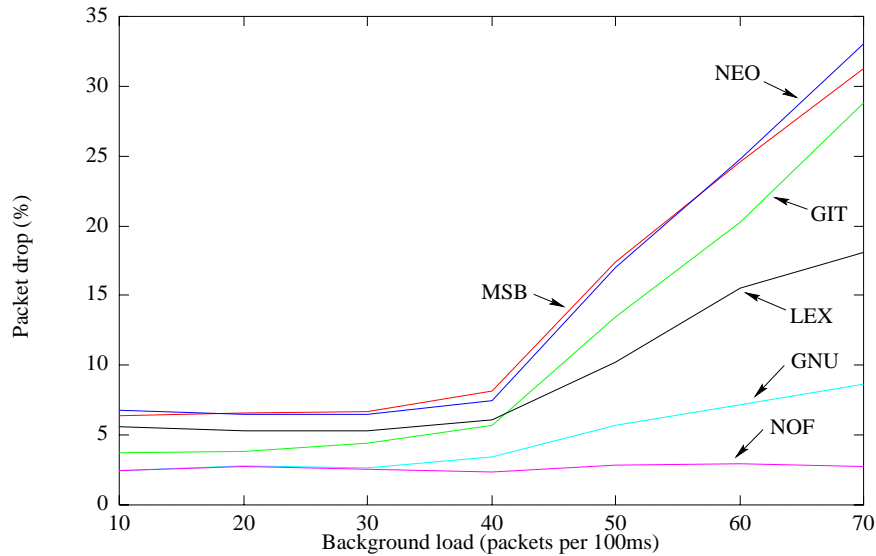


Figure 3.24: SURAnet round-trip drop rates for  $\overline{M} = 10 - 70$  background packets per 100 ms.

#### 3.10.2.2 vBNSnet performance

To achieve a one-way network drop rate of 20% in the vBNSnet network model described in Section 3.1 requires  $\overline{M}$  at each NAP to approach 60000 packets per second. At this packet rate, a five hundred second network simulation requires 450MB of real memory, and three days execution time on an UltraSparc 170 equipped with 320MB RAM. To bring the memory requirement below 200MB and execution time less than 24 hours, the packet size distribution shown in Figure 3.12 is scaled by a factor of three to 4500 bytes. Although a 4500 transmission unit exceeds the maximum sized Ethernet packet of 1500 bytes, such a maximum transmission unit (MTU) is reasonable in future high speed networks. Even today, 4500 byte packets are common on the vBNSnet backbone since many end-systems are connected via FDDI (note that the MTU in an FDDI network is 4500 bytes.)

The vBNSnet background load  $\overline{M}$  is parameterized between 700 and 2700 packets per 100 ms. Tables B.5 and B.6 in appendix B give the simulated vBNSnet round-trip delay

### 3.10. SURAnet and vBNSnet Simulation Parameters 67

for  $\bar{M} = 700$  and 2700 respectively. Like the SURAnet simulation, the round-trip times under light utilization ( $\bar{M} = 700$ ) are almost entirely composed of WAN propagation and campus network delay. However, unlike SURAnet, delays under peak utilization ( $\bar{M} = 2700$ ) increases slightly. This is because the transmission and queuing delays in a high-speed OC-3C infrastructure become insignificant as compared to the propagation delay. Figure 3.25 plots the observed delays from the SFC node to selected sites. Again, note the slow increase in round-trip delays as compared to the SURAnet environment.

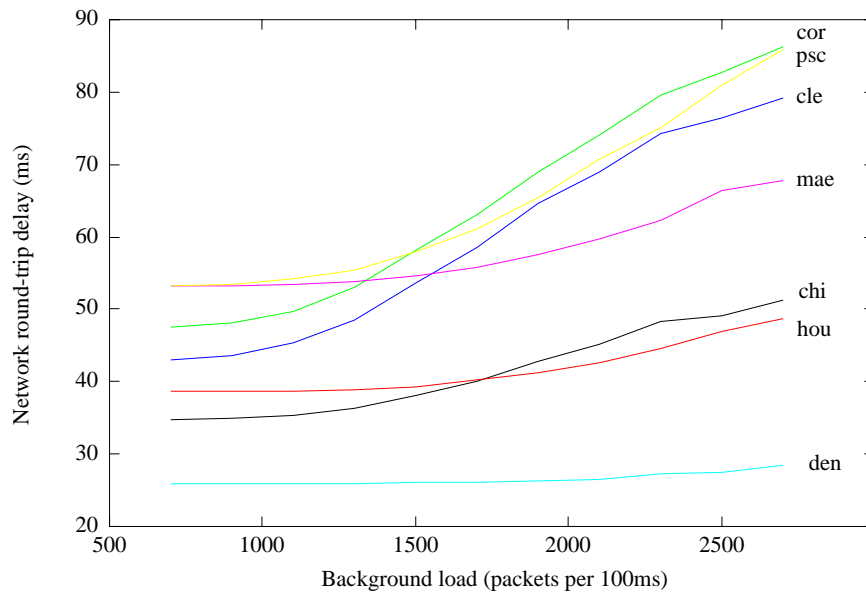


Figure 3.25: vBNSnet round-trip delays for  $\bar{M} = 700 - 2700$  background packets per 100 ms.

Tables B.7 and B.8 in appendix B give the vBNSnet round-trip drop rates for  $\bar{M} = 700$  and 2700 respectively. Under light utilization ( $\bar{M} = 700$ ), the round-trip network drop rates are between 2 – 4%, and peak at 38% under high utilization ( $\bar{M} = 2700$ ). Note that like SURAnet, as the network load increases, bottleneck points begin to appear. In the vBNSnet simulations, the bottleneck appears in the CLE and CHI NAPs. Thus, network performance on either side of the bottleneck points is still good at high utilization. However,



performance across the bottleneck points degrades sharply with  $\overline{M}$ . Finally, Figure 3.26 plots the observed network drop rate from SFC to selected sites. Note the poor performance to CLE and COR as compared to PSC; PSC's route follows SDS, HOU, and MAE and avoids the CHI bottleneck.

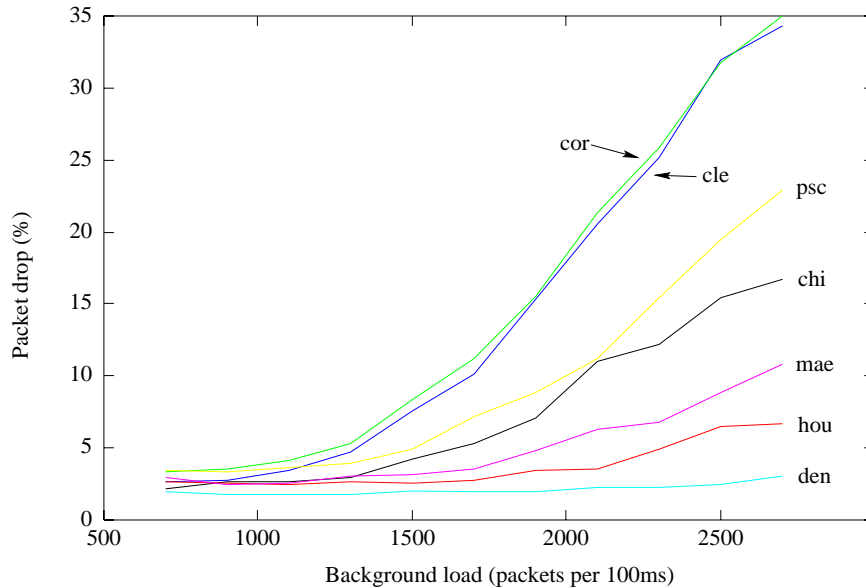


Figure 3.26: vBNSnet round-trip drop rates for  $\overline{M} = 700 - 2700$  background packets per 100 ms.

### 3.11 Summary

In this chapter, we developed a high-fidelity network simulation environment suitable to evaluate the performance and overhead of multicast transport protocols. We considered two backbone networks, namely SURAnet and vBNSnet. SURAnet is chosen because it represents a contemporary low-speed network using T1 infrastructure, whereas vBNSnet chosen because it represents a state-of-the-art network based on OC-3C infrastructure.

Both the SURAnet and vBNSnet networks interconnect large campus networks modeled after UVAnet.

Next, we developed traffic models suitable to drive network performance characteristics of SURAnet and vBNSnet. Within the campus network, we use empirical and SNMP performance studies of UVAnet to drop and delay packets independently at each router. The wide-area background traffic model, however, is considerably more complex. Here, we first consider the statistical properties of ten day, wide-area packet trace. Using arrival density and correlation analysis, we developed the  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  traffic model, which works as follows: First,  $\mathcal{M}$  uses a sophisticated self-similar traffic model to create a sample path representing the aggregate stream generated by a large campus network. Next,  $\mathcal{P}$  partitions the aggregate stream into substreams based on target density functions, while preserving the correlation structure of the aggregate stream. Finally,  $\mathcal{S}$  uses a packet train approach to model short-term dependencies.

We showed that  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  accurately and efficiently generates background traffic by comparing the synthetic streams to the empirical traces used in the characterization study. Using traffic traces and operations reports, we developed  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  traffic parameters which achieve network drop rates from 2 – 20% on the worst-case SURAnet and vBNSnet network paths. In subsequent chapters, we use the SURAnet and vBNSnet environments to implement and evaluate large-scale multicast transport protocols.

## MESH-R: Large-Scale, Reliable Multicast Transport

---

Research in advanced transport services for multicast applications has accelerated in recent years with the continuing maturation of network-layer multicast support. Central to transport protocol designs is the control structure which distributes protocol processing, localizes error repair, and aggregates receiver state. Further, the control structure determines the flow of end-system control information used by feedback-based algorithms, e.g., congestion and error control protocols at the source. Thus, the design of control structure is critical since it determines the services that can be provided at the transport layer, as well as its scalability, efficiency, and performance.

One common control structure is to organize multicast group using a *logical structure* such as a ring, central site, or tree. Tree-based schemes, in particular, offer efficient aggregation of receiver state and low latency error recovery when the control tree closely follows the underlying multicast routing tree. Constructing and maintaining efficient control trees, however, is difficult without resorting to manual configuration, network-layer support beyond current multicast services (e.g., *subcast*), or network-specific options (e.g., hop-count scoping) that have unpredictable effects. Alternatively, some protocols multicast control information to the entire group. SRM [41], for example, uses a sophisticated timer man-

agement strategy based on propagation delay estimates to control access to the multicast channel. Since transport services are not localized, unstructured approaches have a higher overhead and a greater error recovery latency, in general, when compared to tree-based schemes.

This chapter presents a novel multicast transport design (MESH-R) that achieves fully reliable data distribution in a scalable, efficient fashion. A key contribution of our solution is the receiver-driven, dynamic organization of the multicast control structure based on network performance characteristics. The MESH-R framework (1) is a fully distributed, transport-layer solution, (2) presents a robust state synchronization protocol that provides detailed end-system state for reliability, congestion control, group management, and other end-system services, and (3) achieves an efficient, low-latency error control service using a self-organizing, soft-state unicast recovery structure between multicast receivers.

This chapter compares the performance of MESH-R relative to other classes of solutions for reliable single-source bulk transfer in an IP WAN environment. To assess relative performance and network overhead, MESH-R and three other approaches (a centralized, a tree-based, and an unstructured protocol) are implemented in a high-fidelity WAN testbed [64, 66]. This environment allows for direct comparisons between the performance (throughput) and efficiency (protocol messages transmitted across the network) of these schemes. The simulation experiments are performed over a range of network loss rates using two different Internet backbone topologies. The results validate correct operation of MESH-R, and show that error control and state feedback heuristics work well. Specifically, we show that MESH-R's error control protocol adapts rapidly to changing network conditions, exhibits good locality in recovering lost packets, and effectively exploits network links outside of the multicast routing tree. Our experiments show that the MESH control structure has the robustness and flexibility of unstructured approaches, while retaining an efficiency and performance of an optimal tree-based scheme.

The remainder of this chapter is organized as follows: Sections 4.1 and 4.2 motivate this research and discuss related work. Section 4.3 motivates MESH-R's design, and gives a detailed explanation of the state feedback and error control algorithms. Section 4.4 describes our simulation-based performance study. Section 4.5 gives our conclusions and discusses important aspects in further improving and understanding MESH-R's performance.

## 4.1 Motivation

Key multicast application domains studied today include (1) delay-sensitive delivery for multimedia streams, (2) single-source reliable data transfer involving large numbers of receivers (e.g., software distribution, push applications), and (3) multi-source data distribution systems (e.g., web-cache updates, DIS, stock-quote dissemination, and networked games). Most of these applications require reliable data delivery, or, perform best when data loss is minimal. Therefore, reliable transport services are beneficial since they facilitate and simplify multicast application development.

This chapter studies protocol mechanisms for achieving large-scale, reliable data delivery in unreliable networks such as the Internet. Although this research considers a single-source, reliable bulk data distribution application, the results generally extend to the applications discussed above. The following design criteria are critical to a modern reliable transport protocol:

*Reliability:* All nodes identified by the group management protocol either receive the message, or the source is informed that particular data elements were not delivered to particular group members. Note that full-reliability represents the most challenging error control requirement. However, the constraint can be relaxed to achieve other reliability semantics (e.g., k-reliable and receiver-reliable).

*Network Dependence:* Only basic connectionless, datagram IP multicast service should be assumed. In particular, we reject non-standard network-layer service such as subcasting or hopcount-based (TTL) scoping. Subcasting is a powerful additional service which allows

IP routers to receive an encapsulated IP datagram, de-encapsulate it, and multicast it down the local routing tree. Thus, subcast messages reach only the receivers within a subtree (below the router handling the subcast) of the multicast routing tree associated with the source. While proposed in a number of forms, subcasting lacks a standard definition and standardization does not appear imminent. Moreover, reliance on this mechanism in any transport solution will be problematic, e.g., heavily switch-based networks where IP routers are sparse. Hopcount-based scoping also has severe limitations as a mechanism for grouping receivers or limiting transmissions to subsets of the global multicast group [102]. Moreover, the performance of the mechanism varies dramatically and unpredictably for receivers in different areas of the network, even for receivers within the same extended LAN.

Our network model does assume the notion of *domain-based scoping*, as supported by IPv6 multicast addresses. This service option defines multicast domain hierarchies at natural network administration domains (e.g., LAN, campus network, regional backbone, and global). The performance study below considers only the global (all members of the multicast group) and campus (members within a set of physically co-located LANs under one administrative entity) domain. Note that even without network domain scope service, campus scoping can be easily achieved by having the group management protocol establish a separate multicast address for each campus.

*Scalability:* Multicast protocol designs must scale to large receiver sets and heterogeneous network environments.

*Robustness:* Network performance characteristics constantly change due to link and network node failures, congestion, and routing changes. Likewise, the group membership might also change since nodes may join, leave, and fail. The transport should not fail as a result of such dynamic events. Instead, the transport protocol must define robust procedures that adapt quickly to changing network conditions.

*Single Source/Multiple Source:* A distinguishing characteristic of a multicast solution is whether it extends multi-source applications. Control structure solutions specifically for the

multiple-source case include k-ary trees [58] and hypercube organizations [61]. Our protocol is evaluated here for single-source reliable transfer only, but it has attractive properties for extension to the multiple-source case.

## 4.2 Related Work

The control structure approach employed by a multicast transport protocols is the key factor which impacts scalability, efficiency, data delivery latency, and the set of services which can be supported. Control structure classes proposed in the literature can be characterized as: *centralized*, *tree-based*, and *unstructured*. Each is briefly discussed below.

*Centralized control (CM)*: The most basic control structure relies on each receiver exchanging control data directly with the source. Efficiency and scalability are gained by combining negative acknowledgments, sender polling, and periodic receiver-driven state synchronization messages.

*Tree-based control (RMTP, TMTP, LBRM)*: Tree-based approaches extend the centralized schemes such that overhead is distributed hierarchically among group members. Receivers unicast control messages to their parent in the control tree structure. Receiver state is aggregated at internal nodes in the control tree as it propagates towards the source(s).

*Unstructured control (SRM)*: In contrast to the centralized and tree-based designs, unstructured approaches do not organize the multicast group into a control structure. Instead, control messages are multicast to the group using a timer-based strategy. For example, upon detection of an error, a group member schedules a repair request sometime in the future based on the group size and delay from the source. Since control information is multicast to the group, other members suppress redundant requests and replies. Sophisticated timer management strategies achieve a low-latency service, while efficiently using network resources.

### 4.2.1 Analysis of Control Structure Classes

Each of the control structures perform well for certain network environments, group sizes, and application requirements. Each control class is discussed below in terms of the evaluation criteria identified in Section 4.1.

*Centralized approach:* Since the source effectively engages in an independent conversation with each group member, the central approaches have a clear simplicity advantage compared to any distributed control scheme. Thus, any reliability scheme (both single and multi-source) is easily accomplished. The disadvantage is that message processing and network overhead increases significantly as a function of group size and network error rates, thus limiting throughput and scalability. Typically, centralized protocols are used for single-source applications with a small number of group members (e.g., less than 100), or in applications where the feedback to the source is minimal.

*Tree-based approaches:* The hierarchical structure used in the tree-based schemes resolves the scalability limitations of the centralized protocols. The critical factor determining the overhead and throughput of tree-based approaches is the degree to which the multicast group can be organized around the underlying network performance properties. For example, consider the SURAnet network architecture shown in Figure 4.1.

Figure 4.1 shows a tree-based structure in which the group is hierarchically organized around the routing tree (shown as a dashed line with a source located at the CTV campus). This control tree leads to efficient use of network resources since the state is aggregated along the routing tree, and retransmissions traverse a minimum number of WAN links. However, a control tree organization that does not map to the underlying routing tree leads to poor latency and increased network overhead. For example, consider a control tree where nodes located on the MSB subnet are children of an UMD node. Such a tree results in significant network overhead since protocol traffic traverses five WAN links as opposed to a single WAN link. Further, transport latency increases since error repairs are not localized and state slowly propagate towards the source.



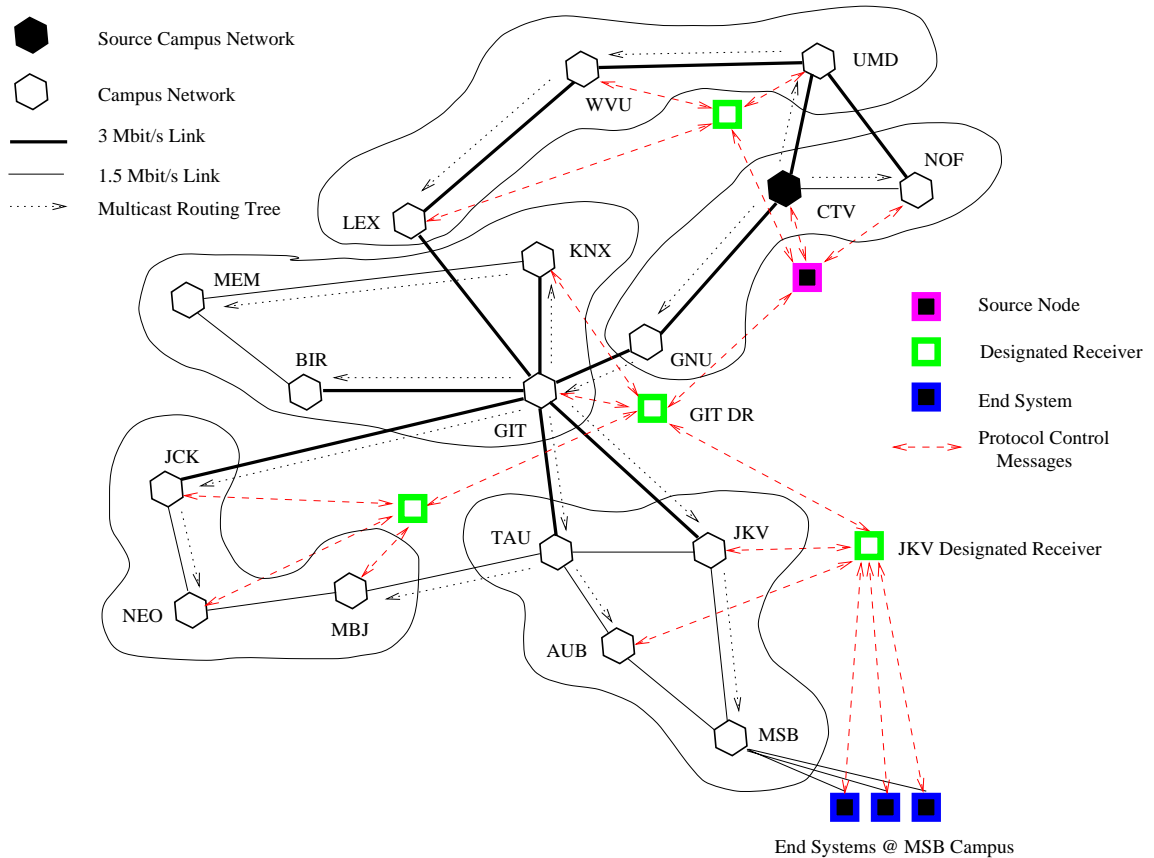


Figure 4.1: SURAnet network with multicast source located at CTV, and group members located at each campus network.

Constructing the control tree such that it maps to the underlying routing tree (or network performance characteristics) represents a considerable challenge without help from the network layer. That is, in a transport-only approach, members of the multicast group can only estimate the transmission latencies using probe packets, and compare relative error characteristics from the source(s). Such information is expensive to collect (i.e., it takes several iterations to estimate the RTT and obtain informative error characteristics), and gives poor information about the routing tree. Consequently, it is difficult to hierarchically

organize the multicast group into a tree structure at the transport layer. This problem is aggravated under dynamic membership (i.e., frequent joins and leaves), when the multicast network has dynamic error and delay characteristics, and the interior nodes of the tree fail (thereby disconnecting the control tree and requiring a repair mechanism).

Given the difficulty of constructing and maintaining a control tree, most protocols rely on help from the network layer to create and maintain the structure of the control tree. For example, RMTP protocol mechanisms are built into network routers. Thus, the control tree is integrated with the multicast routing tree giving excellent performance. Likewise, the TMTP protocol uses TTL-based scoping mechanisms to discover relative network-router distances between nodes and the source. Based on the hop-counts, receivers can be approximately organized to the underlying routing tree.

With help from the network layer (or off-line manual organization), tree-based protocols perform exceptionally well. However, the network-level support currently employed by the tree-based protocols (i.e., TTL scoping, subcasting, and other router-based auxiliary mechanisms) will not be available in future network environments. This is mostly due to the shift from router-based network transport to switch-based WANs using ATM, Frame-Relay, tag-switching, IP switching, and switched Ethernet and FDDI in the local/campus domains. In a switched network, multicast messages are forwarded at the link layer in hardware and are not processed by IP software. Thus, auxiliary IP-level mechanisms will not be available. Manual organization is equally inappropriate for short-lived applications or in dynamic environments.

*Unstructured Approach:* As opposed to relying on network support to help organize the multicast group, unstructured approaches send control messages on the multicast channel and use a timer-based back-off strategy to suppress redundancy. The unstructured approach is attractive for multi-source applications. It also has desirable robustness property in that any node can fail and the multicast session continues without expensive failure recovery mechanisms.

### 4.3. MESH-R Protocol Design and Motivation 78

The key disadvantage to the unstructured strategy is that it becomes increasingly difficult to set the timer back-off strategy in large-scale, heterogeneous network environments. This is because the transmission delays between nodes are significant, thus requiring large timer intervals to avoid duplicate control messages. Consequently, service latency increases. Performance and overhead also suffer significantly in networks with high error rates. For example, in the ideal case only a single retransmission request and retransmission would be generated for each network drop. However, without a local-recovery mechanism, retransmission requires a multicast to the entire group. Consequently, network overhead increases quickly for large groups in networks with even a moderate error rate. This problem is further exacerbated when retransmissions are lost.

## 4.3 MESH-R Protocol Design and Motivation

Section 4.2 characterized multicast transport protocols based on their respective control structure. We saw that there are essentially two strategies – unstructured and highly structured approaches (e.g., centralized and tree-based). This section describes and motivates the MESH-R protocol. We show that MESH-R sits between the unstructured and highly structured approaches, having the robustness and network independence of the unstructured approach, yet the efficiency and performance of a structured approach.

### 4.3.1 MESH Framework Overview and Motivation

The MESH framework localizes protocol overhead by partitioning the multicast group along network domain boundaries. The advantage to the domain structure is that nodes within a domain experience similar error rates, delays, and network performance relative to a source. Further, the domain structure provides a natural partitioning given today’s network architecture (e.g., LAN/campus boundary, campus/regional WAN boundary, regional/backbone WAN interconnect boundary).

### 4.3. MESH-R Protocol Design and Motivation 79

For simplicity and clarity of explanation, this research considers two network domains: campus and WAN. For example, consider the SURAnet network shown in Figure 3.1. Multicast messages sent by a node can be scoped locally to their campus domain, or globally to all nodes<sup>1</sup>. We further assume a group membership protocol which has identified each group member and the campus domain to which each member belongs.

Once the group hierarchy is established, two fundamental issues remain: state synchronization and error control within a domain, and state synchronization and error control between domains. Section 4.3.2 presents MESH-R's state synchronization protocol; Section 4.3.3 presents MESH-R's error control protocol.

#### 4.3.2 MESH-R Group State Synchronization Protocol

In general, the group state synchronization mechanism is a fundamental transport service since it is used by the (1) source(s) to track the state of the group (i.e., which messages have been received), (2) error control mechanism to detect errors, (3) congestion control algorithm to determine network performance and congestion, (4) group management protocol to determine which nodes are alive and their status, and (5) by auxiliary mechanisms to estimate round-trip times, error patterns, and more. In terms of reliable data distribution, the goal of the state feedback service is to (1) provide efficient and low-latency feedback to the source and other group members identifying which nodes have received which data, and (2) provide auxiliary mechanisms that produce an accurate characterization of network drop rates and transmission delays.

MESH's group state synchronization protocol uses a two-part strategy. The first strategy, intra-domain state synchronization, relies on each node multicasting domain-scoped state messages. This approach has several desirable properties. First, each member can identify and track the state of other members within their domain. Second, each node

---

<sup>1</sup>Section 4.5 discusses how the domain structure can be hierarchically extended to scale to multicast groups distributed across global networks such as the Internet.

### 4.3. MESH-R Protocol Design and Motivation 80

can determine network delay, error, and other characteristics between members. Third, any node can compose an aggregate state message representing all the nodes within the domain.

MESH's second synchronization strategy is to elect a single node within each domain to act as the domain's "active receiver" (AR). The key role of the AR is to forward domain state to the next domain in the hierarchy. For example, consider Figure 4.2 which shows four members ( $A, B, C, D$ ) in the same network domain. Here, each member sends a domain-scoped (Dcast) state report in response to receiving data from the source. At some point, the domain AR (in this example  $A$ ) globally multicasts (Gcast) an aggregate state report message representing the state of ( $A, B, C, D$ ).

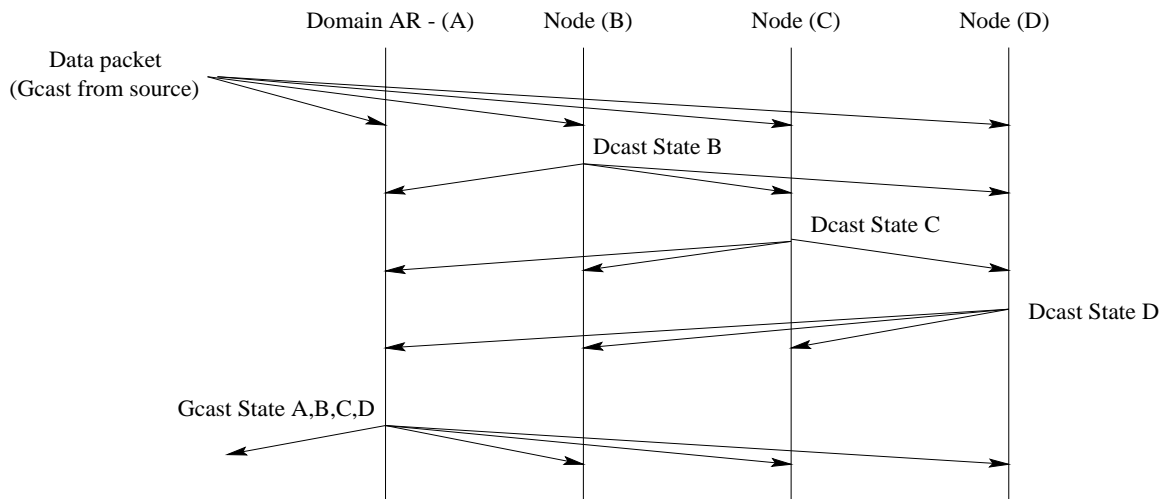


Figure 4.2: MESH's state synchronization approach.

Note that since state is reported via domain-scoped messages, any node can function as an AR. ARs can be determined by a simple election algorithm, or by the group management mechanism. Exactly what state is contained within state messages, and the degree of synchronization, depends on the application's requirements. These details, in the context of a single-source reliable application, are discussed in the next section.

### 4.3. MESH-R Protocol Design and Motivation 81

Node Identifier
Time-stamp
Message Type (Data, Retransmission Request, State Report)
Data Format (if Data Message)
Sequence Number
Size
DATA
Retransmission Request (if Retransmission Request Message)
Sequence Number
General State Report
$C$ – Cumulative DATA ACK
$H$ – Sequence number of highest DATA message received
Loss Mask[ $C, H$ ] – Which DATA messages received over $C \dots H$
Domain Report
$C_D$ – Cumulative DATA ACK domain received
$H_D$ – Sequence number of highest DATA message domain received
Loss Mask[ $C_D, H_D$ ] – Which DATA messages the domain received over $C_D \dots H_D$
Original Received Report
$L$ – Start sequence number of report
Loss Mask – Which DATA messages originally received over $L$

Figure 4.3: MESH-R message format.

#### 4.3.2.1 MESH-R State Report Message Format

Figure 4.3 shows the format of a MESH-R message. The packet header gives the unique identifier of the node (for IP multicast networks, this would be the IP address + port number of the application), local time-stamp of when the message was sent, and message type. Data messages (either original data from the source, or a retransmission) include a sequence number, message size, and the data. Likewise, retransmission requests include a sequence number identifying which data is to be retransmitted.

Each MESH-R message includes a “general state report,” “domain report,” and “original received report.” In the general state report,  $C$  identifies the sequence number of the

### 4.3. MESH-R Protocol Design and Motivation 82

cumulative data messages received by the reporting entity. Next, “loss mask” identifies which messages have been received on the interval  $[C, H]$  ( $H$  gives the highest message received thus far.) Likewise  $C_D, H_D$  gives the cumulative and received bitmask indicating which messages have been received within the domain (if this node is an AR). Finally, the “original received report” indicates the messages originally received in the domain without being dropped by the network.

#### 4.3.2.2 MESH-R State Report Frequency

MESH’s state feedback approach incurs significant network overhead since state messages are multicast to every group member in the domain. Therefore, one goal is to minimize the number of state messages generated by each group member. On the other hand, data throughput depends on the latency of the feedback loop identifying which messages have been received by each group member. The state reporting mechanism must balance the network overhead and performance tradeoff.

MESH-R uses a general state reporting approach suitable for a number of application classes, including reliable multicast. The approach is based on two state reporting strategies. The first is a low frequency, synchronous report generated at interval *SYNC*. Secondly, asynchronous reports are generated when new data arrives, or the state of the domain changes (if the node is an AR).

Synchronous reports serve two purposes: (1) they provide a heartbeat which informs other nodes about this node’s existence, and (2) they update ARs and the source in the event an asynchronous report is lost. Synchronous reports are generated only if an asynchronous report has not been generated for a period of *SYNC* ms. The *SYNC* interval is determined dynamically to be the worst-case latency between any two nodes within the domain.

Asynchronous reports are generated for newly received data (either from the source, or as the result of a retransmission). Asynchronous reports serve the following three roles: First, they act as a positive acknowledgment to the AR (or source). Second, remote nodes

use the reports to cancel a pending retransmission (since the retransmission is no longer necessary.) Third, asynchronous reports are used by the error control heuristic (discussed next) to determine where to send retransmission requests.

### 4.3.3 MESH-R Error Control Protocol

In the MESH framework, error control is based on a unicast request/response retransmission strategy. Unlike the structured approaches, MESH does not build an explicit recovery tree. Instead, MESH relies on a number of heuristics to localize error control based on RTT estimates between receivers and network error patterns observed. Like the state feedback protocol, the MESH error recovery protocol first tries to locally recover lost messages within the domain. For those packets which cannot be recovered locally, the AR attempts to recover the message in the next higher domain.

The state diagram for MESH-R's error control protocol is shown in Figure 4.4. As the figure shows, nodes detect network losses by a gap in the data stream, or from state messages. If data loss is detected either from out-of-order messages in the stream, or from a sender state report, a retransmission request is scheduled immediately. On the other hand, if data loss is detected from another node's state report, a retransmission request is scheduled after the one-way transmission latency from the source. In the later case, the retransmission request is delayed because it is possible that data is still in transit from the source; it may be that the remote node's state report arrived before the data. This time-out prevents unnecessary retransmission requests and responses. Finally, notice that if out-of-order data or a sender state report arrives before the timer expires, the retransmission request is immediately scheduled.

The MESH-R server selection algorithm identifies a remote node to unicast a retransmission request. If the remote node has the data, it immediately unicasts a retransmission, otherwise it queues the request until the data becomes available. After a retransmission request is sent, the node waits twice the estimated round-trip time to the selected server



### 4.3. MESH-R Protocol Design and Motivation 84

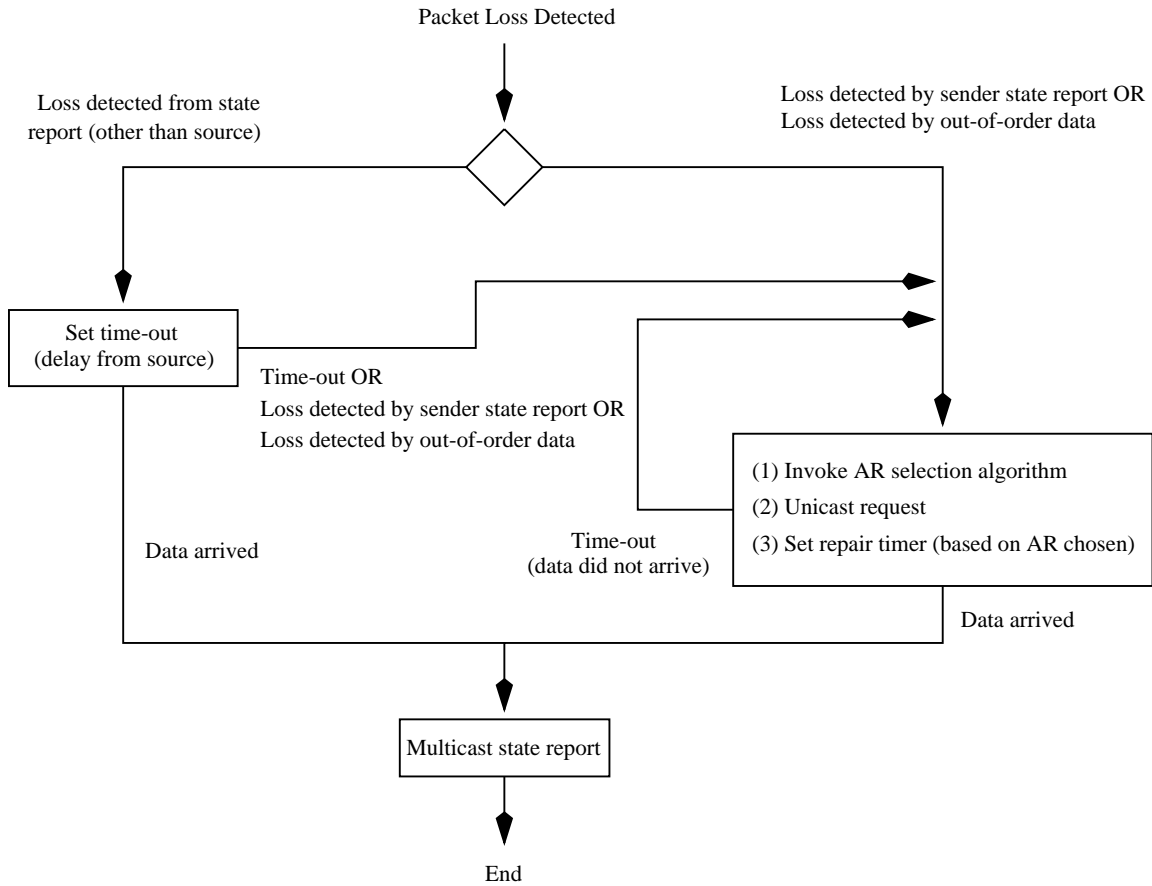


Figure 4.4: MESH-R error control protocol state diagram.

before sending another retransmission request. The time-out period is selected for the following reasons: First, it gives the server node time to recover the message itself in the event the data is not immediately available for retransmission. Second, it provides extra time in the event that current network delays are higher than previously observed. Finally, the delay gives the state reporting mechanism time to identify nearby nodes in the domain which have the data (this is used by the server selection algorithm, discussed next).

#### 4.3.3.1 MESH-R Retransmission Server Selection Algorithm:

In the MESH framework, group members are responsible for initiating and recovering lost data. As shown in Figure 4.4, each node invokes the MESH-R's server selection algorithm to decide where to send a retransmission request. The server selection algorithm is designed using the following goals:

- **Centralized default:** When the network error characteristics and transmission delays are not established (e.g., during initialization) or there is insufficient error characterization available, the protocol should behave like a centralized protocol (i.e., ARQs should be sent to the AR or source as opposed to random behavior).
- **Localized recovery:** Once the state feedback service has collected network error and delay characteristics, error recovery should be further localized to nearby, upstream nodes within the domain (similar to tree-based recovery.)
- **Soft-state, robust, dynamic structure:** The error control structure should adapt to dynamic network performance characteristics. Further, if a node should fail or drop out of the multicast group, then the error control algorithm should quickly route around the failed node.
- **Cross-link/redundant link recovery:** The selection algorithm should take advantage of “cross-link” recovery in networks with a rich/redundant topology. For example, consider the architecture and routing tree for a source located at the MAE-EAST domain in the vBNSnet architecture shown in Figure 3.2. The SFC and DENVER domains should exchange error control information since each node is at the end of a different path from the source, yet both nodes are in close proximity. Likewise, if SDSC has a high error rate (and thus, so does SFC), but SFC is quickly recovering messages from DENVER, then SFC should be able to repair SDSC's errors as well (i.e., downstream recovery.)

### 4.3. MESH-R Protocol Design and Motivation 86

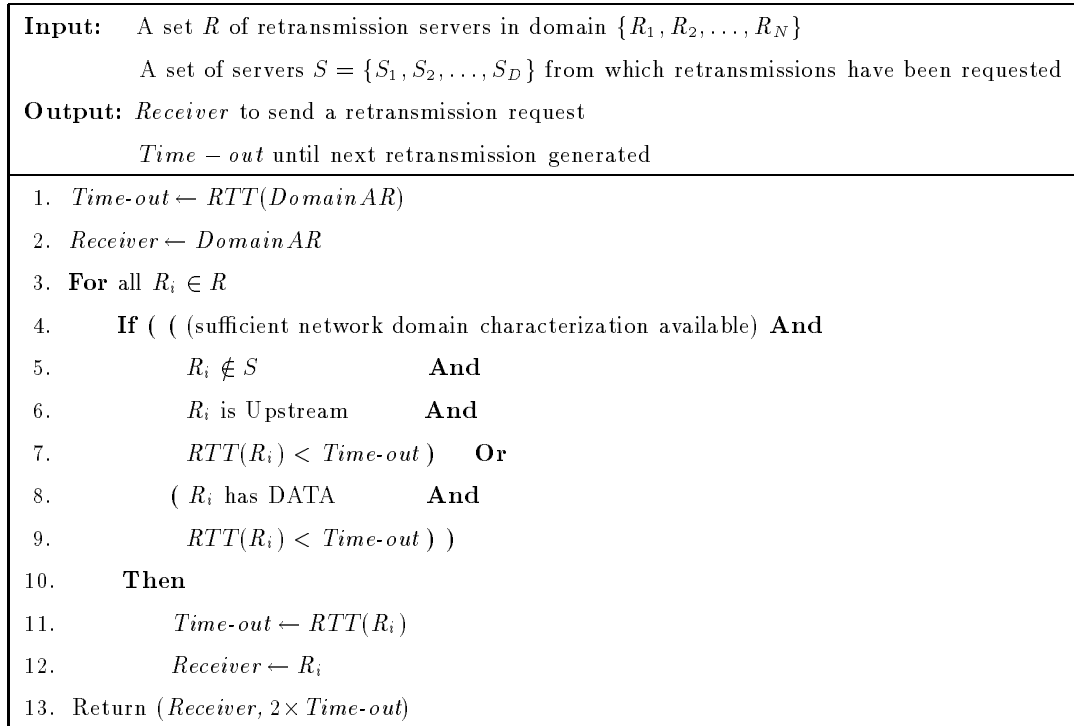


Figure 4.5: MESH-R procedure for selecting a retransmission server.

The MESH-R server selection algorithm is shown in Figure 5.3. The algorithm takes as input a set of group members within the domain (obtained from the state feedback service), and a set of nodes in which an ARQ request has been sent for the missing data. By default, lines 1 and 2 select the domain AR (or source, if the source is located within the domain) to send the retransmission request. Hence, in the default case, MESH-R within the domain behaves as a centralized protocol. Since domains are hierarchically organized around the network infrastructure, MESH-R, by default, behaves similarly to a tree protocol with the hierarchy defined by the network domains.

The goal of the MESH-R retransmission server selection heuristic is to identify nodes that can service the request with lower network overhead and latency than the domain AR. The MESH-R heuristics localize error recovery using a two-part strategy. The first

### 4.3. MESH-R Protocol Design and Motivation 87

phase (given in lines 4-7) considers all the nodes within the domain (line 3). As shown in line 5, a node is eliminated if a retransmission request has already been sent to that node (from the previous retransmission round). Next, a node is eliminated unless it is considered “upstream” to the source, where upstream is defined as follows:

*Upstream( $X, Y$ ):* Relative to a particular source,  $X$  is considered “upstream” from  $Y$  if the number of messages originally received at  $X$  but not at  $Y$  is greater than the number of messages originally received at  $Y$  but not at  $X$ .

Thus, if  $X$  is “upstream” it is a good choice to send a retransmission request because there is high probability  $X$  will have the missing data. Note that the originally received messages are determined by the “originally received message report” component of a state advertisement. Finally, line 7 selects the node that is “upstream” with the lowest estimated network delay.

The second phase of the server selection algorithm is given in lines 8 and 9. Phase two optimizes the basic server selection heuristic by considering all nodes in the domain regardless of whether an ARQ has been sent there. Line 8 reduces this set to those nodes that are known from the state feedback service to have the missing data. Line 9 selects the node if it is closer than the node selected by the first phase of the selection algorithm.

If the data is not recovered within two RTTs to the AR, then the MESH-R heuristic is invoked again to select another AR to send a retransmission request. Note, however, the nodes to which an ARQ has been sent are eliminated from phase one of the server selection algorithm. Hence, MESH-R approximates a “search ring” which expands up towards the source based on error characteristics. On the third recovery attempt, MESH-R considers the network error characterizations to be in a state of flux, and the domain-AR is selected as the retransmission server.

#### 4.3.3.2 MESH-R AR Recovery

The AR is a special node in the error recovery process because it is responsible for recovering data which can not be recovered locally within its domain. Therefore, when an AR detects missing data, it waits to see if any subgroup member reports that it has the data. By default, the AR waits for a period equal to the observed worst-case round-trip time within the domain. If the AR does not receive a domain state report indicating a node has the data, it attempts to recover the data in the higher domain in the hierarchy using the MESH-R recovery algorithm. After the data is recovered, the AR domain-multicasts a retransmission to the other members in the domain.

#### 4.3.3.3 MESH-R Server State

When a node receives a retransmission request, it first checks to see if the data is present. If the server node has the data, it immediately unicasts a retransmission to the requesting node. If the data is unavailable, the server queues the request until it recovers the data via its own error recovery attempts. Once the data is recovered, the server forwards the retransmission to requesting nodes unless the state reporting mechanism indicates the requesting node has obtained a copy of the data.

After sending a retransmission (or after the receipt of a global retransmission from an AR or source), a node sets a “retransmission block” timer. During the “block” time the server node ignores retransmission requests for the data. The “block timer” eliminates retransmissions that would otherwise be generated by a request that was in transit prior to the arrival of the retransmission.

## 4.4 MESH-R Performance Analysis

This section uses the SURAnet and vBNSnet simulation environments developed in chapter 3 to evaluate the MESH, centralized, tree-based, and unstructured protocol designs. Perfor-

mance is evaluated by considering file transfer times and network overhead for a large-scale, bulk data distribution application. Specific goals of the study include:

- Demonstrate the correctness of MESH-R’s error recovery heuristics and state feedback service.
- Evaluate the effectiveness of the AR selection heuristics (e.g., localized and cross-link recovery).
- Demonstrate expected file distribution performance for each protocol class.
- Compare file distribution latency and network overhead for each protocol class over a range of network utilization.

#### 4.4.1 Experiment Setup

Chapter 3 presented the packet-level simulation environment in detail. To summarize, two network models are developed: vBNSnet (OC3 infrastructure) and SURAnet (T1 infrastructure) which interconnect campus networks (Ethernet with FDDI backbones<sup>2</sup>). Network drop rates and delays are fixed within campus networks, and are driven by the  $(\mathcal{M}, \mathcal{P}, \mathcal{S})$  traffic model within the WAN. Key simulation and application parameters are discussed below:

- Background load: Within SURAnet, the background load is varied from 10 – 70 packets per 100 ms, and from 700 – 2700 packets per 100 ms in vBNSnet. These background load rates were chosen because they create a network loss rates between 1% – 25% on the worst-case WAN path.

---

<sup>2</sup>Refer to Figures 3.2,3.1,and 3.3 for the vBNSnet, SURAnet, and campus network infrastructure and topologies.

#### 4.4. MESH-R Performance Analysis 90

- Receiver set: Twelve group members are uniformly distributed at each campus network (i.e., four receivers are located on each Ethernet segment). Thus, there are 204 group members in the SURAnet environment and 132 members in vBNSnet.
- Group management: Receivers are identified before data transfer begins, and do not leave the group during the session. Since reliable data transfer is a requirement, session throughput is limited by the slowest receiver.
- Multicast source: The multicast source is arbitrarily located at the CTV campus in SURAnet, and MAE-EAST within vBNSnet.
- File transfer sizes: A file size of 1 MB is chosen in SURAnet, and 10 MB in vBNSnet.
- Flow control: In the SURAnet simulation, the application's buffer size is fixed at 60 KB at each source/receiver. Data is transmitted using a packet size of 1 KB, therefore a maximum of 60 packets can be in transit at a given time. In the vBNSnet environment, the buffer size is set at 240 KB and data is packetized in 4 KB messages. A window-based approach is used for flow-control. Thus, the source can not send a data message whose packet sequence number is greater than 60 beyond the sequence number of the lowest packet acknowledged by the entire group.
- Congestion control: Typically, the transport protocol adjusts the size of the flow control window as a function of observed network congestion (typically measured by loss rate). However, changing the window size will impact the throughput and network overhead differently for each protocol class. Therefore, the size of the flow control window is fixed throughout the data transfer.

#### 4.4.2 Centralized, Tree-based, and Unstructured Protocol Descriptions

This section defines the centralized, tree-based and unstructured protocols considered in the performance study. Each protocol is designed to give representative performance for

its respective control structure class. No protocol optimizations were made for the network environment, group size, or application under consideration; rather we focus on fundamental designs.

#### 4.4.2.1 Centralized Protocol

The centralized protocol design (similar to [53]) is intended to represent a simple, centralized approach to achieving reliable data transfer. Illustrated in Figure 4.6, data transfer proceeds as follows: The source transmits data on the global multicast address until the transmit window is full (i.e., 60 packets are outstanding). The source keeps track of when each data packet was multicast to the group. Upon receipt of a data message, a receiver unicasts an ACK to the source. The ACK message contains a cumulative acknowledgment (C-ACK) indicating that all data messages up to C-ACK have been received and a bitmask indicating which packets have been received beyond the C-ACK.

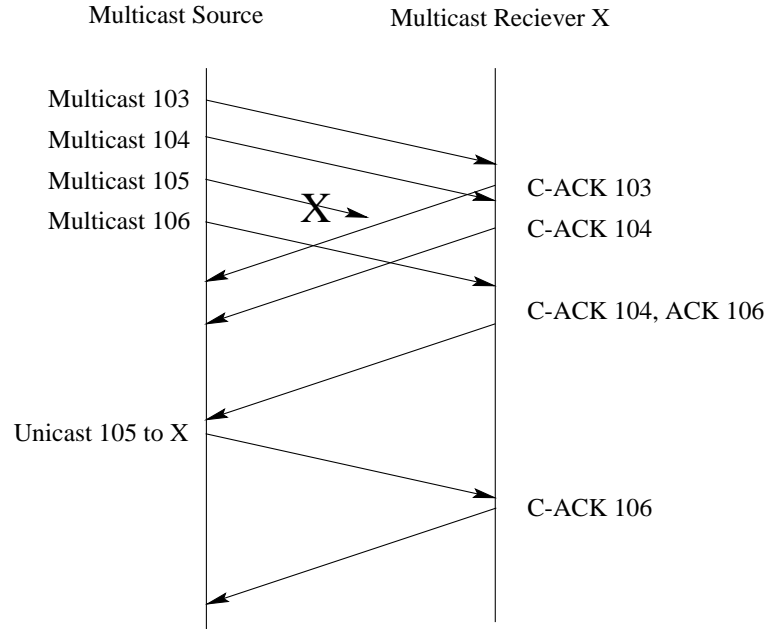


Figure 4.6: Centralized error/control flow example.



#### 4.4. MESH-R Performance Analysis 92

If the source does not receive a positive ACK for a group member after twice the estimated round-trip time (or receives an ACK from a subsequent data message indicating data is missing), the source unicasts a retransmission and resets the transmit time to the group member. For example, Figure 4.6 shows the network dropped data packet 105 to receiver  $X$ . Upon reception of data packet 106, the receiver generates a cumulative ACK of 104, and a selective ACK for 106. Based on the ACK, the source determines  $X$  is missing data unit 105. Therefore, the source unicasts a retransmission of 105 to  $X$ . Also note that if packet 106 was dropped as well, the source would eventually timeout and retransmit both 105 and 106 to  $X$ .

##### 4.4.2.2 Tree-based Protocol

The tree-based protocol is similar to the centralized approach, except that error control and state feedback are localized between nodes in the hierarchy. For example, Figure 4.7 shows the source multicasting data unit 100. Like the centralized protocol, each receiver (this example shows only four receivers:  $A, B, C, D$ ) immediately unicasts a C-ACK(100) to its parent node. Each parent node unicasts a group-ACK (G-ACK) to the next parent in the tree after it determines that all child nodes have received the data. For example, node  $A$  unicasts G-ACK(100) to its parent node after receiving C-ACKs from leaf nodes  $B, C$ , and  $D$ .

Like the centralized protocol, parent nodes use ACKs to determine if child nodes are missing data. Upon detection of a lost data message (as determined by a ACK message, or a time-out<sup>3</sup>), the parent node retransmits the data via unicast to the child node. Like the centralized protocol, parent nodes track when each transmission was sent, and continue to generate retransmissions until an ACK is received from the child.

---

<sup>3</sup>The time-out is chosen to be the estimated round-trip to the child node from the time the parent node received the data from the source.

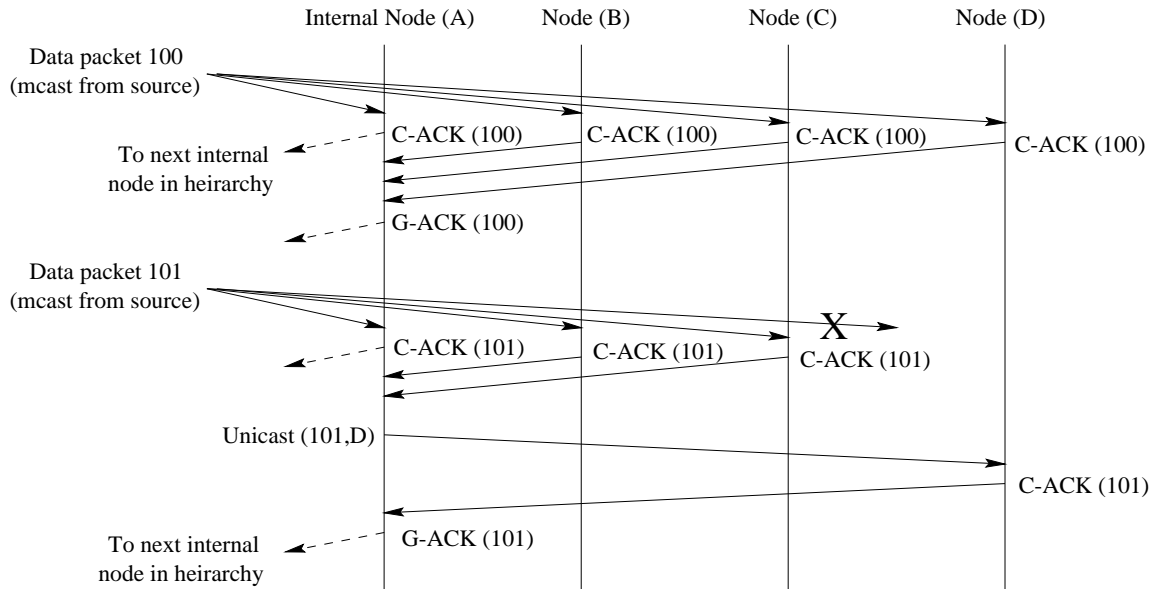


Figure 4.7: Tree-based error/control flow example.

In addition to ACK messages, parent nodes unicast low-frequency polling messages to each child node containing the highest received data sequence number thus far. Each child node replies to a polling message with an ACK. The polling mechanism synchronizes state in the event no ACKs were received from the child node within the polling period (either because the source did not generate any data during the polling period, or the last ACK from a child was lost).

Control trees are statically configured according to the multicast routing tree (thus representing the optimal control tree). In both the SURAnet and vBNSnet simulations, a single node is established within each campus network to act as the campus parent node. Thus, most data messages dropped within the campus network are recovered locally. Campus parent nodes are then hierarchically organized according to the multicast routing tree. For example, consider the TAU node in the SURAnet environment. Here, TAU's parent is GIT, whose parent is GNU, whose parent is CTV (the source). Like the control tree, the polling

period is statically determined depending on the network architecture. A polling period of 500 ms is chosen in SURAnet, and 50 ms in vBNSnet.

#### 4.4.2.3 Unstructured Protocol

Unstructured transport protocols multicast data, state, and error control messages to the entire group. Hence, the key protocol design issue involves developing request/response timing strategies which minimize redundant messages while achieving low service latency. In terms of error control, the two key timing issues involve (1) when to multicast a retransmission request and (2) when to send a repair. In this study, we use the widely accepted error control timing-strategy pioneered in SRM [41].

In SRM, after a node (denoted here as  $A$ ) detects data loss it schedules a retransmission request on the uniform distribution:

$$[C_1 d_{S,A}, (C_1 + C_2) d_{S,A}] \quad (4.1)$$

Where  $d_{S,A}$  is node  $A$ 's one-way estimated transmission delay from the source ( $S$ ). If  $A$  receives a retransmission request (for the same data) before its own retransmission request timer expires,  $A$  reschedules its request timer using a random exponential back-off on the interval:

$$2^{i+1} [C_1 d_{S,A}, (C_1 + C_2) d_{S,A}] \quad (4.2)$$

where  $i$  gives the cumulative number of duplicate requests received thus far<sup>4</sup>. When node  $B$  receives a repair request which it can service (i.e., node  $B$  has a copy of the data),  $B$  schedules a retransmission reply on the uniform distribution:

$$[D_1 d_{A,B}, (D_1 + D_2) d_{A,B}] \quad (4.3)$$

If  $B$  receives a retransmission reply before its timer expires,  $B$  cancels its reply. Otherwise,  $B$  multicasts the repair. Due to the probabilistic nature of the request/repair strategy,

---

<sup>4</sup>In our analysis,  $i$  is given a maximum value of 4.

$B$  ignores repair requests for a period of  $3d_{S,B}$  seconds after sending or receiving a repair  $R$ . The “block” period prevents a station from responding to requests that have been serviced by  $R$ .

The parameter  $C_1$  determines the request timer delay, and  $C_2$  determines the request timer interval. Therefore, smaller values of  $C_1$  and  $C_2$  decrease the delay before a request is generated (thus reducing recovery latency.) However, larger values  $C_1$  and  $C_2$  contribute to lowering the number of duplicate requests (thus reducing network overhead.) The same argument can be applied to  $D_1$  and  $D_2$  since these parameters determine the interval over which retransmissions are sent.

Since retransmissions are multicast to the group, one goal is to minimize the number of responses – thus arguing for a large value of  $C_1$  and  $C_2$ . However, response time is also critical, thus arguing for smaller values of  $C_1$  and  $C_2$ . In SRM,  $C_1, C_2, D_1, D_2$  are adjusted dynamically as a function of both the recovery delay and number of duplicate requests and repairs. The adaptive algorithm for  $C_1$  and  $C_2$  works by calculating the exponentially weighted average of duplicate retransmission requests ( $avg\_dup\_req$ ) and average request delay ( $avg\_req\_delay$ ) at node  $N$ :

$$avg\_dup\_req = (1 - \alpha)avg\_dup\_req + \alpha dup\_req \quad (4.4)$$

$$avg\_req\_delay = (1 - \alpha)avg\_req\_delay + \alpha req\_delay \quad (4.5)$$

where  $dup\_req$  gives the number of duplicate requests received and  $req\_delay$  gives the request delay within the previous recovery. The algorithm for adjusting  $C_1$  and  $C_2$  is given in Figure 4.8. First, the algorithm determines if the number of duplicate requests is greater than 1 (line 1). If so, then the request delay ( $C_1$ ) and interval ( $C_2$ ) is increased thereby decreasing the probability of future duplicate requests. Lines 4 – 8 decrease the request interval and request delay if (1)  $avg\_dup\_req$  is less than 0.5 (line 4), and (2)  $avg\_req\_delay$  is less than the round-trip transmission delay from the source (line 5) or if  $avg\_dup\_req$  is greater than 0.25 (line 7) respectively. Decreasing the request delay and

Initial Values
$C_1 = C_1 = 2$
$D_1 = D_2 = \log_{10}(\text{Group Size})$
For each recovery period:
1 If ( $avg\_dup\_req > 1$ )
2     Increase request timer delay ( $C_1 + = 0.1$ )
3     Increase request timer interval ( $C_2 + = 0.5$ )
4 Else If ( $avg\_dup\_req < 0.5$ )
5     If ( $avg\_req\_delay > RTT(\text{Source})$ )
6         Decrease request timer interval ( $C_2 - = 0.1$ )
7     If ( $avg\_dup\_req > 0.25$ )
8         Decrease request timer delay ( $C_1 - = 0.05$ )
9     Else
10     Increase request timer delay ( $C_1 + = 0.05$ )

Figure 4.8: SRM adaptive timer algorithm.

request interval reduces the delay over which a request will be generated, thus improving recovery delay. Finally, if  $avg\_dup\_req$  is between 0.5 and 1.0, the request delay is increased slightly – thereby decreasing the probability of a duplicate request. The same adaptive algorithm is used to adjust  $D_1$  and  $D_2$ , based on the measurements  $ave\_dup\_repair$  and  $ave\_repair\_delay$ .

In the SRM protocol framework, each group member sends a periodic session message which contains the highest sequence number received from each source. In the data-distribution context, session messages are modified to include a C-ACK and a bitmask indicating which messages are received beyond C-ACK. Typically, session messages are generated periodically such that session message bandwidth is limited to 5% of the bandwidth consumed by data and retransmission overhead. In our implementation, session messages are generated by each group member on 250 ms intervals in SURAnet, and 25 ms intervals in vBNSnet. The session message frequency was chosen so that state is propagated quickly to the source without consuming excessive bandwidth.

### 4.4.3 Experimental Results

The performance of each protocol is evaluated using three metrics: application throughput, network overhead, and protocol efficiency. Each metric is defined and used to compare relative protocol performance across a range of network load in SURAnet and vBNSet. Following the preliminary analysis, the performance of each protocol class is discussed in detail.

#### 4.4.3.1 Application Performance

The first metric used in the comparison study considers application performance in terms of elapsed file distribution time. Figures 4.9 and 4.10 plot transfer times for each protocol as a function of network load in SURAnet and vBNSnet respectively. Each data point gives the average transfer time based on ten simulation runs, with error bars showing the range of transfer times observed.

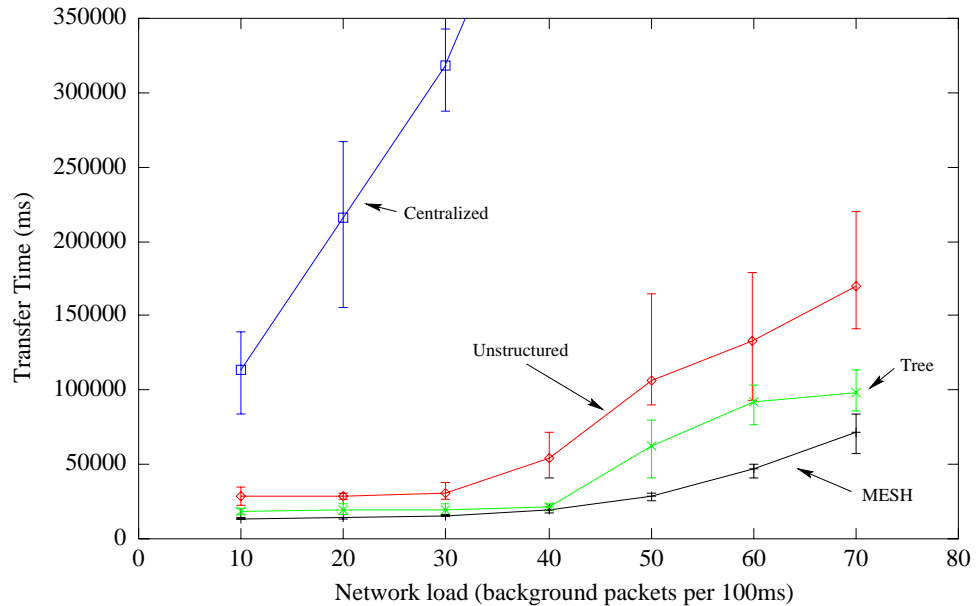


Figure 4.9: File transfer times for 1 MB file in SURAnet.

#### 4.4. MESH-R Performance Analysis 98

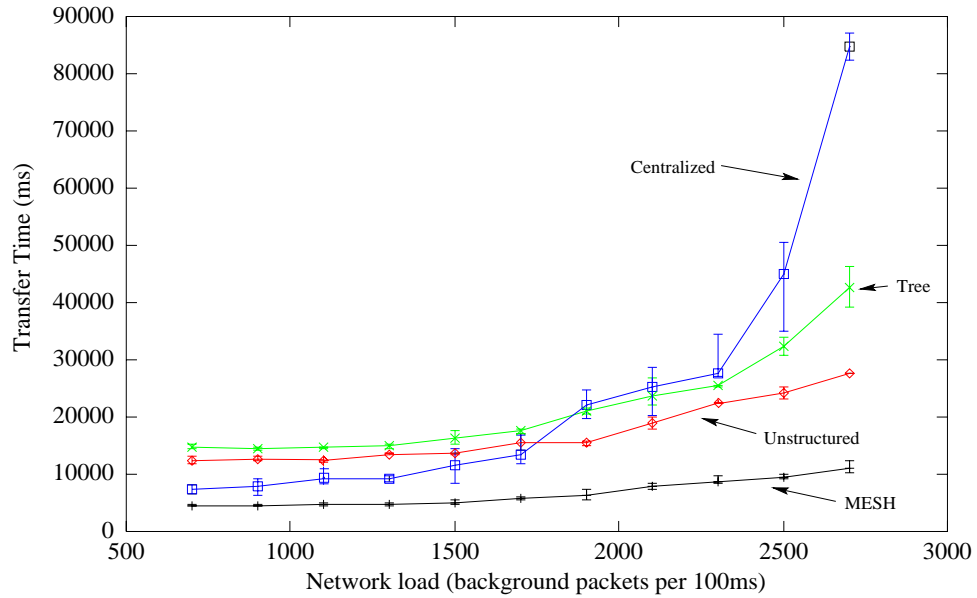


Figure 4.10: File transfer times for 10 MB file in vBNSnet.

Figure 4.9 shows that in the SURAnet environment, the transfer times using the centralized approach were the largest and increase rapidly with network load. In contrast, the measured file transfer times were more than ten times faster using the distributed approaches. Between the distributed approaches, Figure 4.9 reveals a significant performance gap. In particular, we see that under light load the tree and MESH protocols perform comparably and distribute the file twice as fast as the unstructured approach. Over the entire dataset, MESH had the best throughput – performing nearly three times better than the unstructured approach, and twice as fast as the tree protocol under heavy network load (i.e., the background load equals 40 – 70 packets per 100 ms.)

Figure 4.10 shows the average file transfer times for each protocol in vBNSnet. In contrast to the SURAnet experiments, the centralized protocol outperforms both the tree-based and unstructured approaches until moderate network load is achieved, at which point centralized performance degrades rapidly. Secondly, Figure 4.10 shows the unstructured protocol distributes the file 20% faster on average than the tree-based protocol across the

entire range of network load. Finally, the figure shows MESH outperformed the tree and unstructured approaches by a factor of three across the entire range of network load.

#### 4.4.3.2 Protocol Overhead and Efficiency

Although application performance is a key performance metric, it does not characterize network overhead introduced by the protocol. In this research, overhead is evaluated by considering the number of protocol bytes introduced on the WAN. Here, *protocol bytes* is defined as the cumulative number of bytes from (1) data messages multicast by the source, (2) state feedback messages, (3) retransmission requests, and (4) retransmission replies traversing each WAN link. Note that protocol overhead on the campus network is not considered since LAN bandwidth is not constrained in our model.

Figures 4.11 and 4.12 give observed protocol overhead in SURAnet and vBNSnet respectively. The figures show that the tree-based protocol introduces the fewest number of bytes in both network environments. In fact, since the control tree is constructed to map directly to the network routing tree, the number of bytes introduced represents a lower bound on overhead<sup>5</sup>. Next, the figures show that MESH introduces slightly more overhead than the tree protocol in vBNSnet, and exhibits substantially more overhead in SURAnet – comparable to the unstructured protocol. Finally, in both network environments, the figures show that the centralized technique introduces substantially more overhead than any of the distributed techniques.

Next, we evaluate protocol efficiency (i.e., combine both performance and overhead in a single metric) by dividing protocol bytes (given in Figures 4.11 and 4.12) by application throughput (given in Figures 4.9 and 4.10). The efficiency metric is useful because it gives protocol overhead relative to its throughput (which we call *normalized overhead*). The efficiency metric can be best understood by considering the following example. Consider

---

<sup>5</sup>Assuming that the network routing tree represents the lowest overhead path from the source to receiver set.



#### 4.4. MESH-R Performance Analysis 100

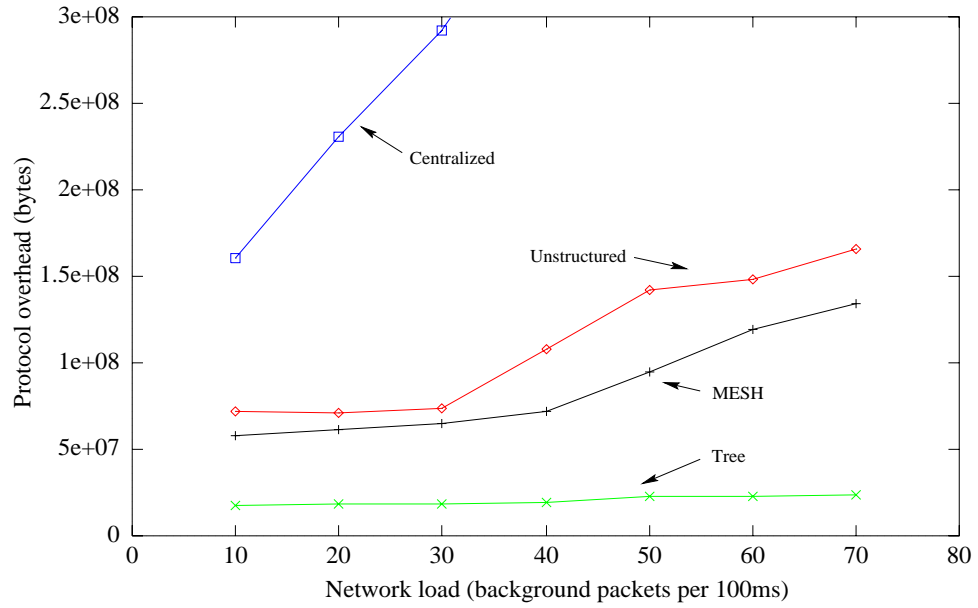


Figure 4.11: Protocol overhead in SURAnet.

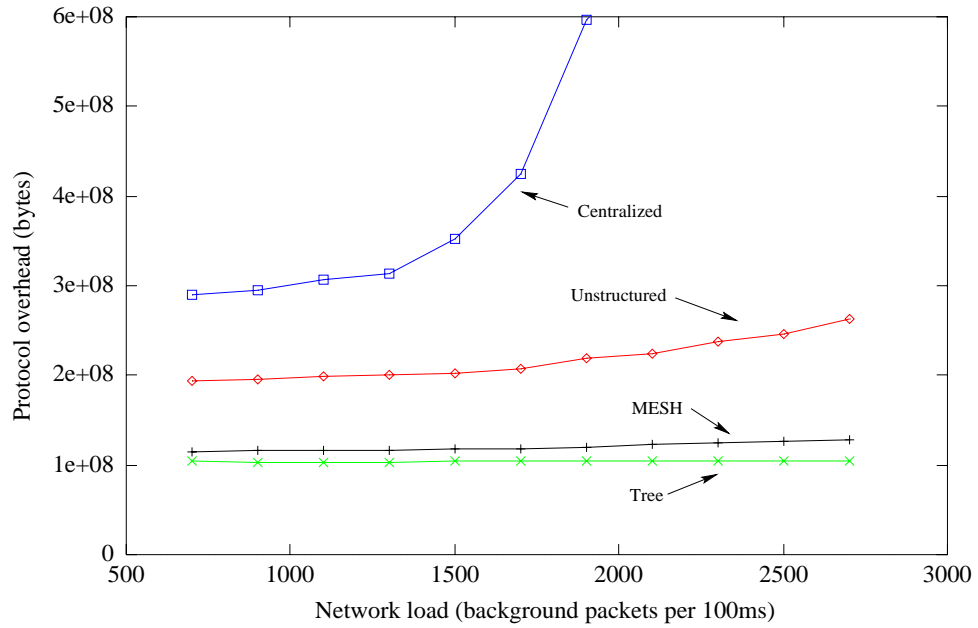


Figure 4.12: Protocol overhead in vBNSnet.

an application using two protocols ( $A$  and  $B$ ) to distribute a 50 KB file. Using protocol  $A$ , the application distributed the file in 50 seconds with  $protocol\ bytes_A = 500$  KB. However, when using protocol  $B$ , the transfer took 100 seconds with  $protocol\ bytes_B = 100$  KB. Here, the normalized overhead of protocols  $A$  and  $B$  are 500 and 200 respectively. Thus,  $B$  is 2.5 times more efficient even though  $A$  transferred the file in half the time.

Figures 4.13 and 4.14 give the normalized overhead for each protocol plotted as a function of the background load for the SURAnet and vBNSnet environments. The data points are based on average file distribution time and protocol bytes for the runs at each background load.

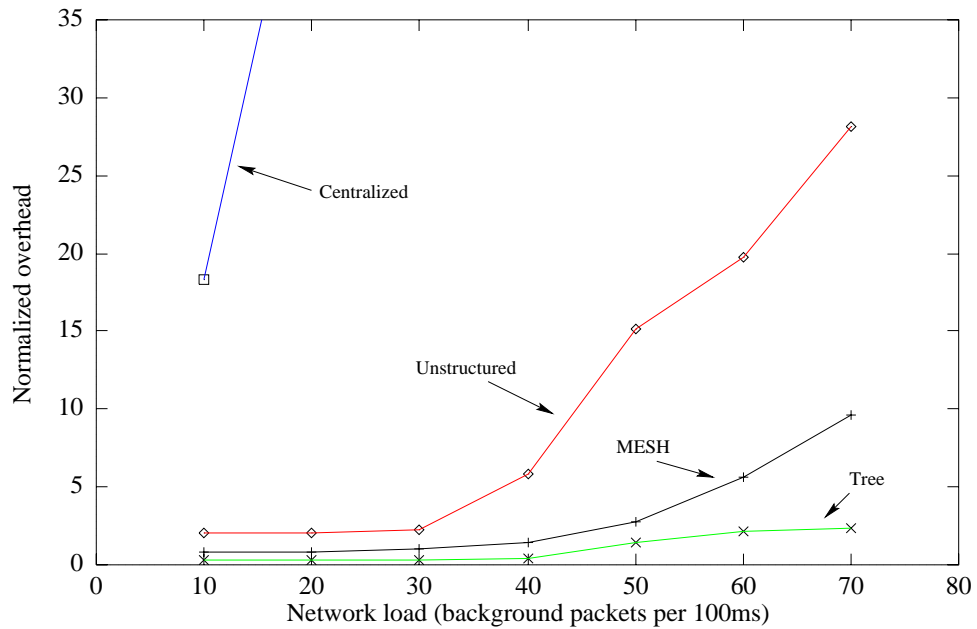


Figure 4.13: Protocol efficiency in SURAnet.

Figure 4.13 shows that the centralized protocol performs very poorly when considering the amount of network overhead incurred to achieve the given throughput. Secondly, unlike the file transfer time metric, there is significant disparity in efficiency between the distributed protocols. Here, the tree protocol performs the best across the entire range of

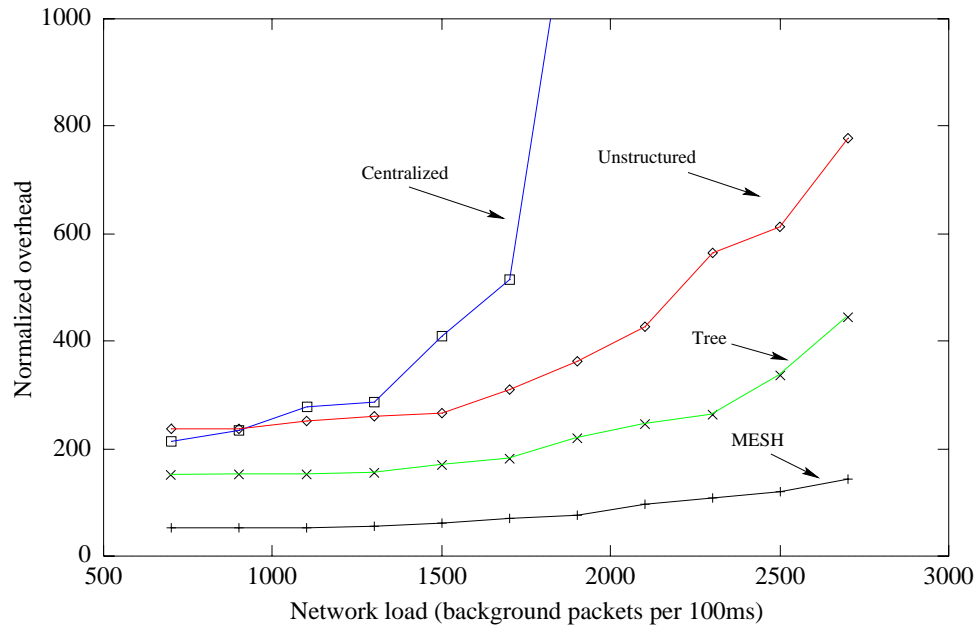


Figure 4.14: Protocol efficiency in vBNSnet.

network load. MESH performs comparably to the tree protocol until high network load, in which case the tree protocol is three times more efficient. Finally, the unstructured approach performs poorly when compared to MESH and the tree-based schemes – particularly at high network load, where the normalized overhead of the tree-based scheme is ten times better.

Figure 4.14 shows similar efficiency results for the centralized and unstructured approaches in vBNSnet. However, the efficiency of the centralized approach is comparable to the unstructured protocol until moderate network load, at which point it increases exponentially. Unlike SURAnet, MESH achieves the best normalized overhead in vBNSnet, outperforming the tree-based scheme by a factor of 2.5 independent of the network load.

#### 4.4.3.3 Centralized Performance Analysis

The throughput and efficiency analyses show that the centralized approach does not scale to large receiver sets in a bandwidth-constrained environment such as SURAnet. In particular, as the network load becomes significant (e.g., background load equal to 40 packets per 100 ms in Figure 4.9), the centralized approach effectively fails without using congestion control (i.e., reducing the window size of 60 KB). The failure is the result of the unicast error recovery overhead (implosion) on network links close to the source. Multicasting retransmissions would lower the overhead, however, not by the order of magnitude necessary to achieve the performance levels of the distributed techniques. The vBNSnet experiments show that if the network can handle the additional overhead, the centralized approach can perform remarkably well. None the less, Figure 4.14 shows that scalability is limited as a result of the exponentially increasing network overhead.

#### 4.4.3.4 Unstructured Performance Analysis

The performance of the unstructured approach is very good as compared to the centralized approaches. However, it performed the worst of the distributed techniques when both throughput and efficiency are considered. The relatively poor throughput performance relates to the effects of dropped requests and responses on the adaptive timer strategy, explained as follows.

Recall equation 4.2 which gave the interval in which a repair request is scheduled after a network drop is detected. For each duplicate request received, a node reschedules its request using an exponential back-off. This approach works well when requests and responses are not dropped by the network (note that in the SRM performance analysis [41], this assumption is made). However, consider what happens when a request is dropped. Some set of nodes missing the data will not receive the retransmission request, and thus will time-out sending its own request. As a result, the adaptive algorithm interprets the duplicate requests as if the two requests crossed paths in the network, and therefore increases the request parameters  $C_1$

and  $C_2$ . Likewise, if a response is dropped, the response parameters  $D_1$  and  $D_2$  are increased. Unless the adaptive timer strategy can distinguish between duplicate requests/responses as a result of network drops, or because they crossed en route within the network, the adaptive timer algorithm will continue to increase the request interval, yielding poor performance.

The second performance limitation with the unstructured approach relates to multicasting retransmission requests and responses. For example, consider the group membership in SURAnet (204 members). During light network loads (e.g., drop rates of 1%), on average at least one node will not receive the data. As a result, the data stream effectively doubles when considering retransmission overhead. When the network drop rate approaches 7%, two retransmissions are sent on average for each message in the datastream – effectively tripling the size of the data stream. This is why Figure 4.13 shows the rapid decrease in efficiency as network drop rates become significant (e.g., at a network load of 40 packets per 100 ms.)

#### 4.4.3.5 Tree Performance Analysis

As discussed in section 4.2, tree-based protocols localize error control and aggregate receiver state between levels in the hierarchy. In our experiments, the control tree was organized along the multicast routing tree, thus is optimal. The experiment data confirms that the tree-based protocol introduced the least overhead in each network.

In the SURAnet environment, MESH introduced 2–5 times the number of protocol bytes as compared to the tree-based scheme. Thus, even though the tree protocol did not distribute the file in the least amount of time in SURAnet, it had the best overhead/throughput ratio. However, in the high-speed vBNSnet network environment the throughput of the tree-based protocol was poor relative to the other distributed protocols. Hence, even though it introduced the least network overhead, MESH significantly outperformed the tree-based scheme when considering the overhead/throughput metric.

The poor throughput performance in the high-speed networks can be explained by considering the latency of the receiver feedback aggregation. For example, consider how state propagates from the NCSR node to the source (MAE-EAST). First, NCSR unicasts a G-ACK to DENVER, which in turn unicasts a G-ACK to CHICAGO, etc. Thus, NCSR's ACK must propagate through a campus network ten times before reaching the source. In contrast, consider the other protocols which send their state directly to the source. Here, the ACK only traverses two campus networks, one at NCSR and MAE-EAST. In a low-speed environment, the campus propagation delay is negligible. However, in vBNSnet, the campus network delays exceed the one-way delay from NCSR to MAE-EAST. As a result, the source must stall waiting on feedback.

#### 4.4.3.6 MESH Analysis

The performance analysis shows that MESH has the highest throughput in both SURAnet and vBNSnet, and the best overhead/throughput ratio in vBNSnet. First, we explain MESH's performance by considering protocol behavior within the campus network. Then we focus on MESH's performance within the WAN.

In the campus network model, packet drops are infrequent and transmission delay is low. Since delays and drops are modeled independently for each message, the MESH heuristics can not determine if nodes within the domain are upstream or downstream. As a result, the MESH selection heuristics default to the domain AR, and therefore behave nearly identically to the tree-based protocol. However, MESH can recover some network drops local to the campus that the tree protocol can not. For example, consider Figure 4.15 which shows a campus network with the AR located on Ethernet A and additional receivers located on Ethernets B and C. The example shows the campus network dropped message 100. Thus, the receivers on A did not receive message 100, but the receivers on B and C did. In this case, the AR detects that a receiver on *B* has the data, and thus recovers the data locally.

In contrast, the tree-based protocol would have sent a request on the WAN (assuming the parent node for the campus network is located on A).

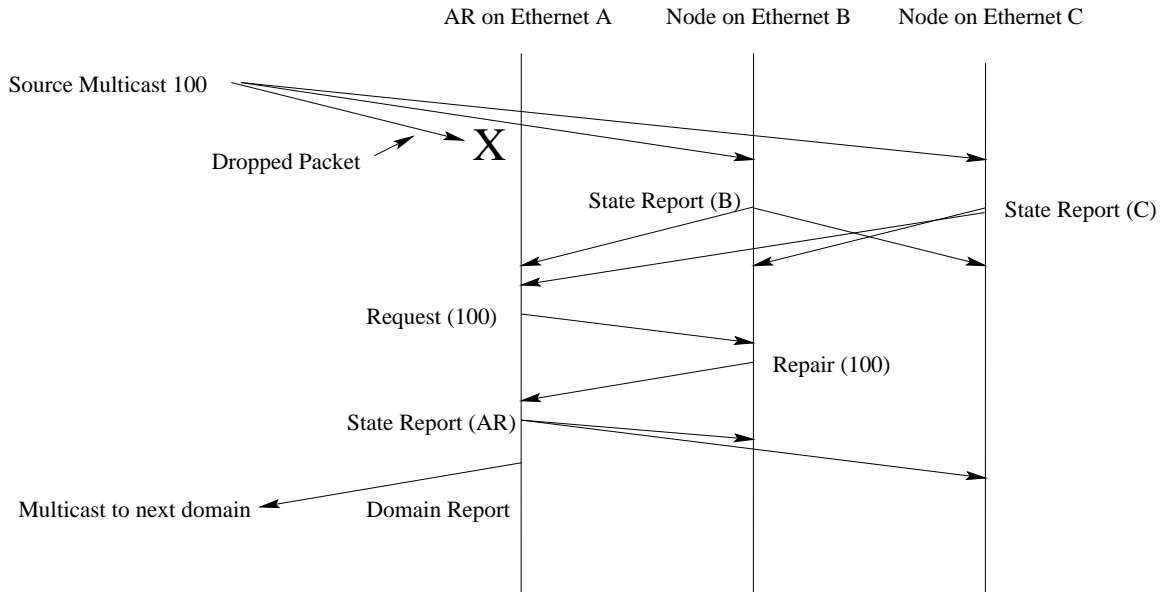


Figure 4.15: MESH error recovery example in campus domain.

Unlike the campus network, error rates and transmission delays in the WAN give the MESH heuristics a rich network characterization to determine where to send retransmission requests. MESH's performance within the WAN can be explained by considering where ARs send their retransmission requests. For example, consider table 4.16 which gives the retransmission request distribution within vBNSnet with background load equal to 2300. Figure 4.16 presents table 4.1 graphically for requests sent between ARs whose campuses are adjacent. Here, the black arrows show the direction of the routing tree and the red arrows show the percentage of retransmission requests sent to the AR located at the adjacent campus.

As Figure 4.16 shows, MESH directed 85% of the retransmission requests to an AR located one WAN hop away. This high degree of local recovery contributed to the low network overhead and recovery delay. In contrast to the tree-protocols, MESH relied heavily

#### 4.4. MESH-R Performance Analysis 107

	sfc	ncsr	sds	den	chi	ncsa	hou	cle	cor	psc	mae
sfc	0	0	77	0	0	0	7	0	0	0	15
ncsr	20	0	1	76	1	0	0	0	0	0	0
sds	12	0	0	0	0	0	84	0	0	0	2
den	87	0	4	0	5	1	1	0	0	0	0
chi	1	0	0	1	0	91	2	3	0	0	0
ncsa	0	0	12	0	6	0	77	0	0	0	3
hou	0	0	0	0	0	0	0	0	0	0	100
cle	0	0	0	0	0	0	0	0	0	83	16
cor	0	0	0	0	0	0	0	72	0	27	0
psc	0	0	0	0	0	0	0	0	0	0	100
mae	0	0	0	0	0	0	0	0	0	0	0

Table 4.1: vBNSnet AR selection distribution.

on cross-link recovery. For example, consider the DENVER node. Here, 87% of the requests did not follow the routing tree to CHICAGO, but rather were sent to SFC. In this situation, MESH chose SFC because it has a significantly greater probability of recovering the data as compared to CHICAGO. This is because the routing path to SFC is independent from the path to DENVER. Likewise, at the CHICAGO node, MESH directed retransmission requests to NCSA. Most of the other nodes, however, forwarded the bulk of their requests to the closest upstream node on the routing tree. As a result, MESH added only 5% more protocol bytes to the network as compared to the tree approaches.

Figure 4.17 shows the percentage of requests directed to adjacent campuses in SURAnet during moderate network load. Like the vBNSnet environment, MESH achieved a high degree of localization – directing 70% of all retransmission requests to an AR one WAN link away. Likewise, MESH exploited cross-link recovery. For example, consider the UMD node. Here, 9% of the retransmission requests were sent up the routing tree to the CTV campus, whereas 88% were sent to NOF. For the UMD node, NOF is a good choice because it has a low RTT as compared to CTV and NOF’s path to the source is independent from UMD’s path. Likewise, other nodes such as LEX, MEM, MBJ, and AUB heavily exploit cross-links properties sending 74%, 97%, 78%, and 79% of their requests off the routing tree respectively.



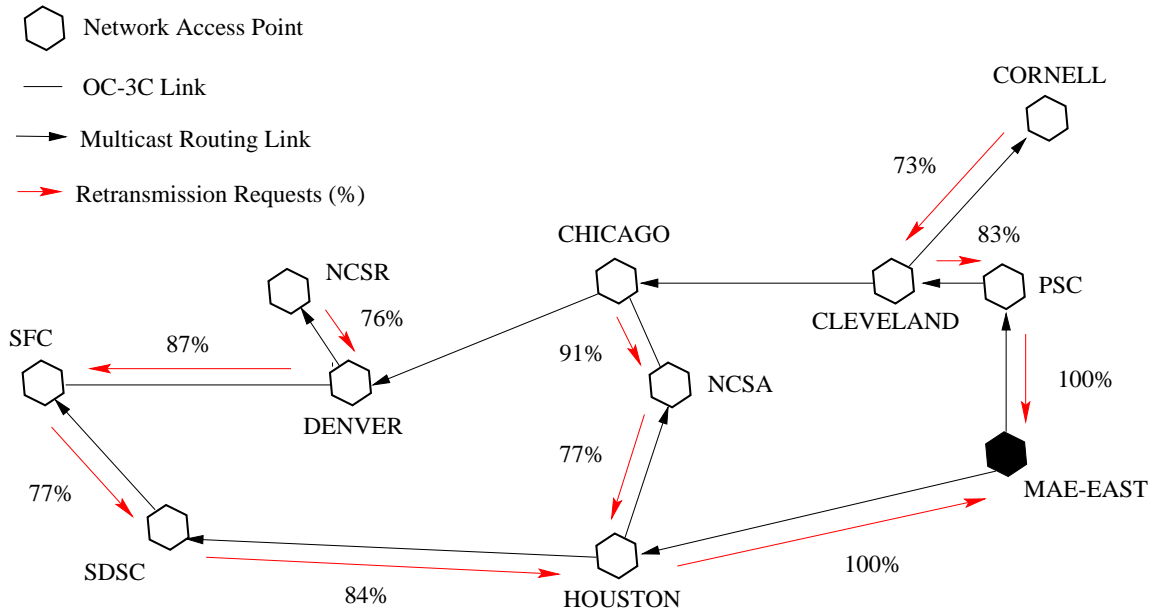


Figure 4.16: Retransmission distribution in vBNSnet between adjacent campus networks.

In addition to cross-link recovery, a few ARs actually sent retransmission requests “downstream” along the multicast routing tree in SURAnet. For example, consider GNU. Even though the source CTV is located a single hop away, the GNU node directed 16% of its requests downstream to GIT. GNU selects GIT because the link between GNU and CTV is a bottleneck point with a significant transmission delay and drop rate (as shown in Tables B.3 and B.4). Since GIT has a lower transmission delay than CTV, GNU selects GIT when GIT reports that it has the data. It is interesting to note that no data packets were dropped on the GNU-GIT link. Thus, GIT can not determine if GNU is upstream or downstream in the routing tree. By default, GIT considers GNU downstream, and thus does not send retransmission requests to GNU.

Although MESH delivers the file quicker than the other protocol classes, Figure 4.13 shows that the tree-based approach outperforms MESH in terms of throughput efficiency in SURAnet. One key factor impacting MESH’s efficiency relates to MESH’s aggressive error

#### 4.4. MESH-R Performance Analysis 109

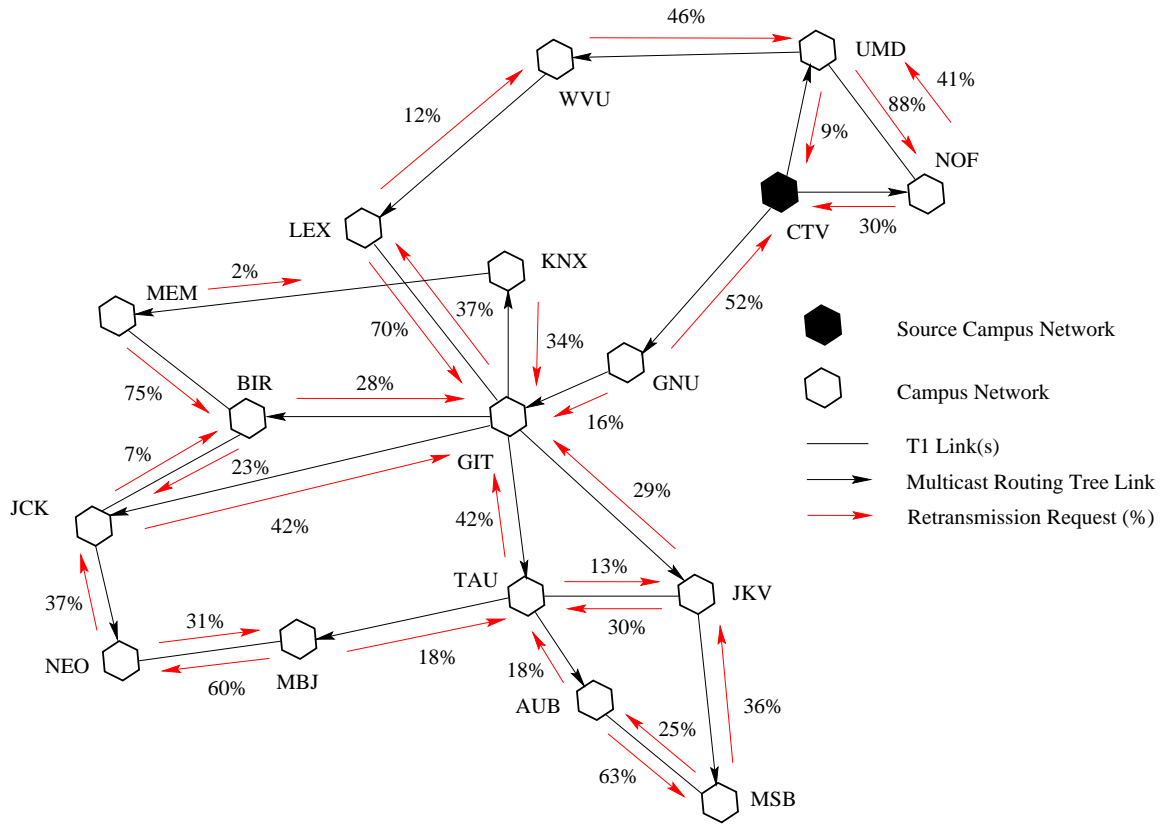


Figure 4.17: Retransmission distribution in SURAnet between adjacent campus networks.

recovery strategy. Recall that an AR waits two RTT periods after sending a retransmission request before it times-out and sends another. For most ARs, the data is recovered on the first try. However, consider the case where a data message is dropped on the Ethernet link at the source. Now, when MSB detects the dropped message it will likely send a retransmission request immediately to JKV. However, since JKV has a low RTT compared to the source, it is unlikely that JKV will have recovered the data before MSB times-out. Consequently, MSB sends a request to the next upstream node, namely TAU. The MSB AR will continue to aggressively search for the data until it is recovered. Using the MESH

strategy, nodes such as MSB often recover the data quicker than when using a tree-based approach, but with a higher network overhead.

The second inefficiency of MESH relates to the state reporting mechanism. That is, each AR multicasts up to two state reports for each message – once when the AR receives a message, and another as a result of the subgroup receiving the message. As compared to the tree and unstructured designs, which only generate a single ACK per message or use periodic strategies, MESH incurs at least twice the state feedback overhead. The next section presents some approaches to reduce the state overhead without compromising throughput.

## 4.5 Summary and Future Work

This chapter presented MESH-R, a reliable single-source multicast protocol built using the MESH framework. Using a high-fidelity network simulation of SURAnet and vBNSnet, the performance of MESH-R, a centralized, a tree-based, and an unstructured protocol was evaluated under a range of network loads. The analysis showed that centralized approaches fail at high network congestion levels, and that distributed protocols are well suited for large-scale multicast networks and groups.

Of the distributed protocols, our analysis showed that the unstructured approach offered the worst overhead/performance ratio. In contrast, we found that MESH-R achieved the lowest file distribution latency in both SURAnet and vBNSnet due to MESH's ability to localize error recovery and exploit cross-link redundancy in the multicast network. Using the network efficiency metric, we found that an optimal tree-based protocol slightly outperformed MESH in the SURAnet environment, and that MESH-R offered the best network efficiency in the high-speed vBNSnet environment.

Our performance analysis shows that the generic implementation of MESH-R is an attractive protocol for the application and network environments considered. In general, however, no two multicast sessions are the same. Each application has different error control and feedback requirements. Likewise, each network has its own particular performance

characteristics. One key feature of the MESH framework is that the heuristics can be programmed for a particular application, network environment, or group size. We conclude by discussing how MESH can be programmed and optimized for various multicast application and network scenarios. In particular, we consider techniques to reduce the overhead of MESH’s state reporting mechanism, improve the performance of the error control heuristics, and adapt MESH to larger multicast network environments.

#### 4.5.1 Future Work on Reducing MESH-R’s State Overhead

In the transport domain, state synchronization is a fundamental service because error control, reliability, congestion control, group management protocols, and other feedback algorithms rely on the service to collect accurate and timely state information. In the MESH framework, synchronizing state is an expensive operation because state messages are multicast to the domain. We are currently investigating techniques to reduce the state overhead in MESH, without sacrificing synchronization latency. Our approach is to characterize application requirements so that MESH can identify critical state (i.e., state impeding the progress of the group), and non-critical state.

As an example, consider the single-source reliable application. The source requires feedback from the multicast group so it can determine which messages have been received by each node. Therefore, the lowest cumulative message sequence number received thus far is the critical state. In the MESH framework, the state synchronization strategy can be driven based on the critical path. For example, consider Figure 4.18 in which the multicast receiver *X* originally received messages 103, 104, 106 – 110. For each message, the local state changed and thus a state report was generated. However, for reliable delivery, the ACKs generated after “C-ACK 104”, were useless until 105 was recovered.

A second optimization is to identify state which is critical to the progress of the entire group, and use that information to drive local synchronization. For example, consider the NOF and MSB nodes in SURAnet. MSB is likely to be a bottleneck point for the group,

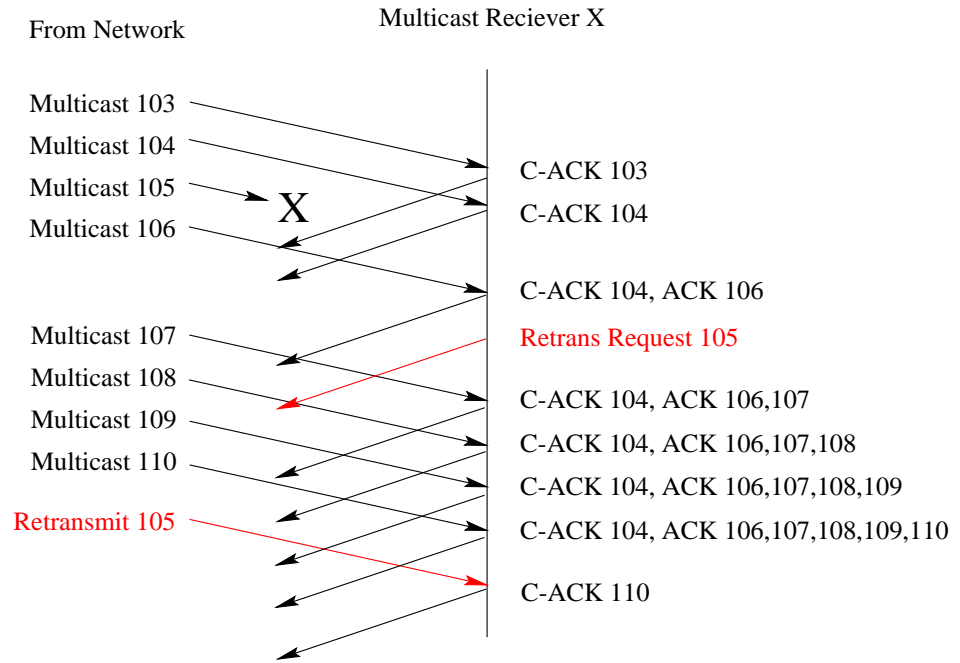


Figure 4.18: Example MESH state synchronization.

whereas NOF is likely to proceed in lock-step with the source. In a similar fashion to the local bottleneck, based on the state of MSB (or the bottleneck node in the group) NOF can back-off its state reporting frequency since it does not speed the progress of the data transfer.

In future work, we will investigate using the critical state techniques to reduce state overhead as well as the aggressiveness of the error control strategy – which can be driven using the same approach.

#### 4.5.2 Future Work on MESH-R's State Synchronization Protocol

The state reduction techniques can lead to substantial efficiency gains for some applications. However, one danger when reducing the state reporting frequency is the effect of dropped state reporting messages. For example, consider the example in Figure 4.18. Here,

the throughput of the group is not impacted if “C-ACK 103” is dropped by the network, because “C-ACK 104” is subsequently multicast. However, if the state reporting algorithm suppresses “C-ACK 104” because it is not considered critical, then data transfer might stall.

We are currently investigating adding a field to the state reporting message (shown in Figure 4.3), which gives the node’s global perspective of the messages received by the entire group. That is, each node reports its view of the group state. This field has several key advantages. First, it allows the nodes in the group to determine the current bottleneck of the group, and thus use it to drive the aggressiveness of the control algorithms. Second, it addresses the problem of dropped state messages. That is, as long as one other node in the group receives the state report, the state will propagate to the other nodes in the system. Finally, the global state field adds significant robustness to the protocol. For example, consider the SURAnet architecture 3.1. Let’s say the CTV-UMD and CTV-GNU links are overloaded to the point where no data messages can cross them. With the global state field, data transfer under MESH-R can continue reliably. That is, NOF can communicate with the source and receive new data. Based on NOF’s state messages UMD will figure out that it is missing data, and recover it from NOF. Eventually, the data will propagate to the entire group. Even though only NOF can directly communicate with CTV, the global state field will indirectly inform the source that all receivers obtained a copy of the data.

### 4.5.3 Future Work on MESH-R’s Error Control

The performance analysis demonstrated that the MESH heuristics do very well localizing error recovery simply by considering network delay and loss characteristics. However, we discovered some examples where MESH was too aggressive sending retransmission requests. In particular, we showed that the MSB node in SURAnet would first attempt to recover the data from JKV. However, approximately 50% of the time JKV could not recover the data within two RTTs. Thus, MSB would time-out and continue its search.

This data suggests that MESH’s error recovery algorithm should adapt to the temporal error recovery pattern in addition to static properties. We are currently investigating a simple adaptive algorithm which drives the time-out delay based on the number of duplicates received. In particular, we are considering a strategy similar to the SRM algorithm which increases the time-out if there is, on average, more than one duplicate retransmission received. Otherwise, the time-out is slightly reduced.

Another strategy we are investigating adapts the time-out based on the state of the node where a retransmission request was sent. For example, if the server node reports it has the data, the time-out can be reduced to a single RTT estimate. This optimizes for the case where the retransmitted data or retransmission request was lost. We believe these adaptive techniques can effectively reduce network overhead without sacrificing performance.

Researchers have shown that forward error correction (FEC) strategies can provide excellent error coverage for large-scale multicast [74, 75]. FEC schemes, however, can not repair all errors. In future work we will consider integrating FEC with the MESH retransmission scheme. In future work we will investigate adaptive FEC overcoding schemes based on the the error pattern reported by MESH’s “originally received” metric.

#### **4.5.4 Future Work on Multi-Source Applications**

Like the unstructured approach, MESH easily extends to a multi-source application. The only modification is that the message format shown in Figure 4.3 would have to carry state for each source. The current state can be easily packed into 32 bytes for each source; thus, the number of active sources must be controlled so that they fit into a single state report message. Note that the error control algorithm does not change. In future work, we will design MESH-W – a reliable transport protocol for multi-source web cache updates, and compare the performance to other large-scale, multi-source applications.

**4.5.5 Future Work on Scalability**

The MESH framework achieves scalability by partitioning the multicast group along network domain boundaries. This research considered two domains, the campus and WAN backbone, with hundreds of receivers. In the campus network model, experimentation showed that the AR could handle 100 nodes with comparable overhead to the 12 receivers studied here. The only limitation is the campus bandwidth and the processing capabilities of the AR. The more challenging domain relates to extending the network hierarchies beyond a single WAN backbone. In future work we will consider the scalability of MESH with three or four backbone domains. In particular, we will investigate the latency of the state synchronization and AR selection within the domain.



## MESH-M – Large-scale Transport for Multimedia Applications

---

### 5.1 Introduction

Demand is steadily increasing for using wide-area multicast networks to deliver large-scale audiocast, videocast, group videoconferencing, and other multimedia applications. Multicasting is needed in order to utilize network resources efficiently and to provide low-latency delivery to multiple receivers. At the same time, for good performance, continuous media data streams such as digital audio or video require bounded transmission delay, low error rates, and predictable throughput. Since multicast networks offering such quality of service (QoS) guarantees are not widely available today, endsystem protocols must be devised to compensate for network delays and errors, else the playback quality at the receivers will be poor.

Buffering on the receiver side protects playback from disruptions due to packet delay variations, *packet jitter*. Receiver feedback techniques [17] are effective in matching application data rates with long-term fluctuations in the available network bandwidth. For network packet loss, redundancy-based error control, or Forward Error Correction (FEC), offers a low latency solution to compensating for some losses but at the cost of consuming

additional network bandwidth [13, 87]. While source-driven retransmission protocols do not scale to large networks and large receiver sets, recent fully reliable multicasting protocols introduce a hierarchy of special multicast receivers that handle retransmissions locally and relay control information back to the source. Distributed schemes offer the advantages of low-latency recovery, restriction of individual retransmissions to only a portion of the network, and scalability to large receiver sets.

This chapter introduces a novel distributed retransmission-based error control protocol for wide-area multicasting of time-constrained data streams. Known as MESH-M [65], the new scheme extends the successful approach of distributed multicast error control in three key ways. First, MESH-M incorporates a retransmission model for delay-sensitive streams, decoupling the use of retransmission from a fully reliable service semantic and allowing the protocol to be tailored for application-dependent delay constraints. Second, the protocol uses a run-time algorithm for dynamic configuration of the control framework between the receivers performing retransmission-based recovery. This dynamic control structure enables adaptation to network traffic dynamics and exploitation of multiple physical paths in the wide-area topology, in contrast with static hierarchical control as in [47, 62]. Third, MESH-M is explicitly designed to work with multiple multicast data sources without building a per-source control framework.

The remainder of this chapter is structured as follows. Section 5.2 covers related work and describes MESH-M. Section 5.3 compares the performance and network cost of MESH-M against end-to-end FEC using a wide-area network simulation of digital video delivery. The simulation results demonstrate that, if the multicast receivers can tolerate some additional buffering delay, MESH-M can substantially improve playback quality by recovering lost packets and that the network overhead incurred by MESH-M is significantly less than with end-to-end FEC. Section 5.4 gives our conclusions and discusses future work.

## 5.2 Distributed Retransmission Error Recovery

Chang and Maxemchuck (CM) [21] present the pioneer work on fully reliable multicast transport protocol design. However, the CM protocol and its derivatives are based on a centralized design which does not scale to large groups [84]. Holbrook [47], Paul [62] and Yavatkar [103] use a distributed protocol design for achieving single source, reliable data distribution. These protocols strategically distribute special nodes called designated receivers (DRs) throughout the multicast group. By hierarchically organizing DRs and endsystems, errors can be recovered locally between DRs and their children. Floyd et al. [41] propose the SRM protocol in which receivers multicast retransmission requests to the group and any group member can multicast retransmissions. The distributed protocols discussed above improve scalability to large multicast groups and wide-area network environments. However, the retransmission approaches employed are largely inappropriate for multimedia applications since the temporal constraints of continuous media are not considered.

A deadline-driven protocol for retransmission-based error recovery of continuous media data is presented by Dempsey [37]. The slack retransmission request (S-ARQ) scheme extends the initial buffering at the receiver (the *slack time*) to enable receivers to recover some lost messages via retransmission. Figure 5.1 [36] illustrates the S-ARQ method for a unicast voice application. In the figure, the network drops the second voice packet in a talkspurt. Upon detecting the lost packet (as triggered by the reception of an out-of-order packet), the receiver initiates a retransmission request and recovers the data prior to its playback deadline.

S-ARQ performs well for point-to-point multimedia applications in a high performance network environment [38]. However, S-ARQ as described above will not scale to large multicast groups because the source must process every retransmission request for the entire group. Furthermore, S-ARQ becomes infeasible in large networks where the roundtrip delay between the receiver set and the source requires a prohibitively large slack time.

## 5.2. Distributed Retransmission Error Recovery 119

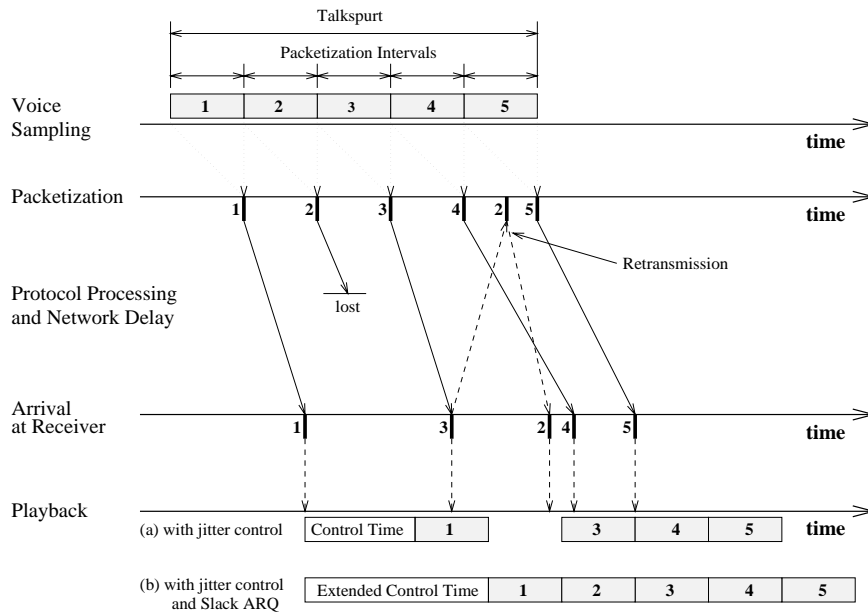


Figure 5.1: S-ARQ retransmission model.

### 5.2.1 The MESH Protocol Framework

MESH-M is a distributed error control protocol for large-scale multicast groups that incorporates the time-constrained retransmission model of S-ARQ. In the MESH framework, the multicast receiver set is partitioned into local subgroups, as motivated by the hierarchical design of contemporary networks: high-performance local or metropolitan area networks with high bandwidth, low error rates, and generally low utilization interconnected by WANs that offer relatively low bandwidth, large delay variations, and high error rates due to heavy utilization. The MESH framework uses various mechanisms for error recovery within the local and the wide-area domain, in order to exploit the high performance of local area networks and to protect scarce wide-area link bandwidth.

### 5.2.2 Error Recovery in the Local Area

Prior to data transfer to a multicast group, MESH-M requires the multicast group to be partitioned into subgroups. The goal of the subgrouping algorithm is to partition the receiver set along the boundaries representing a high performance network domain, or *campus network*, within the multicast network. Each subgroup elects a special receiver, the *active receiver*, which coordinates error recovery for the subgroup. Algorithms for effective subgrouping and electing the active receiver are not explored in this research, but domain-based, centralized, or local subgroup-based multicast schemes present several options.

Active receivers recover losses inside the local campus network via local communication with subgroup members. Traditional source-based reliable multicast techniques are effective in this domain. For losses that occur in the wide-area network delivery, the active receiver recovers packets using the mechanisms described in Section 5.2.3 and then forwards the recovered packets to all receivers in the local subgroup.

### 5.2.3 Error Recovery in the Wide Area

For error recovery between subgroups, the ARs use a novel combination of control mechanisms. During the multicast data transfer, each AR dynamically discovers neighboring ARs with which it may perform retransmissions. Discovery takes place through an *advertisement protocol* run at each AR, thus avoiding the overhead of setting up and maintaining static control hierarchies, e.g. as in [103]. For non-local losses, an AR runs the *retransmission protocol* and unicasts a retransmission request to a remote AR. If an AR receives a retransmission request and has a copy of the data, it immediately unicasts the data to the requesting AR. As with S-ARQ, buffering at the multicast receivers is scaled to ensure a high likelihood that retransmissions can be completed before the playback deadline of the retransmitted packets. Retransmissions arriving after the playback deadline are ignored.

### 5.2.4 Advertisement Protocol

After multicast data transfer commences, ARs send low-frequency *advertisement messages* to a multicast group address on which all ARs listen. Advertisement messages (1) announce the existence of the AR to other ARs, (2) characterize network errors between the AR and each multicast source, and (3) provide a means to estimate the transmission delay between ARs. Note that the advertisement protocol can be easily implemented using an existing protocol definition, e.g., RTP [92].

The structure of an advertisement packet is shown in Figure 5.2. In the header the *Source Identifier* uniquely identifies the AR by concatenating its network address and port number; the *Timestamp* field contains a fine-granularity (e.g., millisecond resolution) timestamp of when the advertisement was transmitted; and the *Report Number* sequences the advertisements from an AR. The body of the advertisement packet characterizes network losses at each AR relative to each multicast source transmitting to the group.

Each block in the body carries four fields. The *Stream Id* identifies the multicast data stream for this block. All information in the other three fields of the block are relative to this multicast data stream. The *S* and *E* fields give the range of message numbers, e.g., packet sequence numbers, being reported. Finally, the *Loss Mask* field is a bitmask over the range [S,E] where a bit set in the *n*th position of the mask represents the correct reception at the AR of the message  $S + n$ ,  $n = 0, 1, \dots, E - S$ . In this context correct reception means that the message was received from the network without any retransmissions.

Active receiver **A** processes the advertisement sent by active receiver **B** in the following manner. First, the network transmission delay between **A** and **B** is estimated via the *Timestamp* value in the header, along with an exponentially weighted moving average for smoothing. Second, the *Loss Mask* field computes an *error independence* metric between **A** and **B**,  $EI(\mathbf{A}, \mathbf{B})$ , relative to each multicast data stream. The error independence metric measures the difference between the network errors experienced by **A** and the network errors experienced by **B**.

## 5.2. Distributed Retransmission Error Recovery 122

Source Identifier
Timestamp
Report Number
Stream Id /* Source 1 */ S /* Start Message Number */ E /* End Message Number */ Loss Mask[S,E] /* messages received in [S,E] range */
⋮
Stream Id /* Source N */ S E Loss Mask[S,E]

Figure 5.2: Advertisement packet format.

Error independence is calculated as follows. For a given multicast data stream, let  $l_X(0, n)$  be a bitmask representing the reception history at active receiver,  $\mathbf{X}$ , for messages with sequence numbers in the range  $[0, n]$ . A bit set in the  $i$ th position represents correct reception of the  $i$ th message. Let the number of bits set in the mask be denoted  $|l_X(0, n)|_{recvd}$  and the number of unset bits by  $|l_X(0, n)|_{lost}$ . Define

$$\chi_i = \begin{cases} 1 & \text{if } |l_B(i, i)|_{recvd} = 1 \text{ and } |l_A(i, i)|_{recvd} = 0 \\ 0 & \text{otherwise} \end{cases}$$

Then, for each multicast data stream common to  $\mathbf{A}$  and  $\mathbf{B}$ , active receiver  $\mathbf{A}$  calculates

$$EI(\mathbf{A}, \mathbf{B}) = \frac{\sum_{i=0}^{\hat{E}} \chi_i}{|l_A(0, n)|_{lost}} \quad (5.1)$$

where  $\hat{E}$  is the largest value received in the  $E$  field relative to this stream within an advertisement message from  $\mathbf{B}$ .

### 5.2.5 Retransmission Protocol

Each active receiver uses the information derived from received advertisement messages to drive retransmission decisions. Specifically, when an AR,  $\mathbf{AR}_{\text{local}}$ , detects missing packets,  $\mathbf{AR}_{\text{local}}$  dynamically determines an appropriate remote AR to which a retransmission request will be sent. The algorithm is given in Figure 5.3.

The three inputs to the procedure are: the playback deadline of the lost packets, the set of ARs from which advertisement messages have arrived, and the set of known ARs which have requested a retransmission of the missing packets. The algorithm loops through all possible candidate ARs to which a retransmission request might be sent and applies the following criteria for selection. First, ARs that have already sent retransmission requests to the local AR are eliminated (line 4) in order to avoid circular retransmission requests. Second, as a heuristic, any AR,  $\mathbf{AR}_i$ , such that  $EI(\mathbf{AR}_{\text{local}}, \mathbf{AR}_i) = 0$  (line 5) is eliminated since  $\mathbf{AR}_i$  is unlikely to have a copy of the missing packets. All ARs that appear to be too far away for timely retransmission (line 6) are removed, and finally, the algorithm biases selection (line 7) towards the AR that will impose the least overhead on the network, as estimated by the network latency from the local AR to the remote AR. If the algorithm returns no candidate for retransmission, then the active receiver suppresses its retransmission request in order to avoid wasting scarce wide-area network resources on retransmission attempts with a low probability of success.

## 5.3 MESH-M Performance Evaluation

In this section, we use a packet-level simulation to evaluate the error recovery performance, network overhead, and scalability of MESH-M. Performance metrics of interest include: (1) effectiveness in improving the data available for timely playback at the receivers, (2) the network overhead of MESH-M protocol traffic, and (3) the distribution of protocol



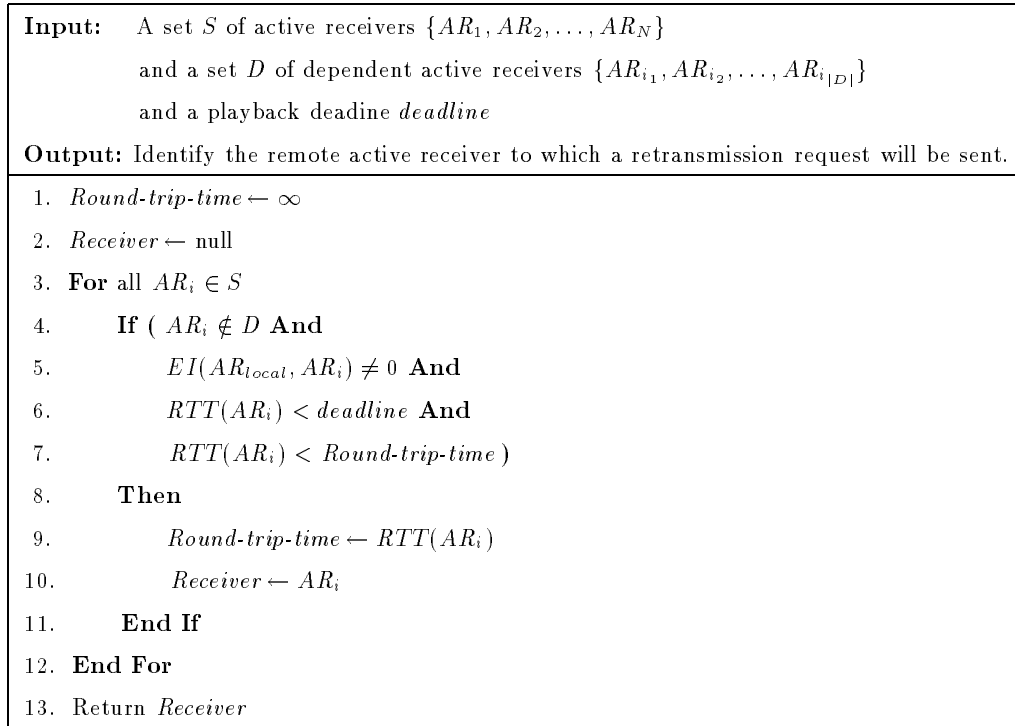


Figure 5.3: Retransmission procedure for selecting remote AR.

processing among ARs. To give a comparison with an alternative solution for delay-sensitive error control, MESH-M is compared against end-to-end forward error correction.

### 5.3.1 Simulation Design

The performance of MESH-M and FEC are compared by considering a videocast application based on the SURAnet simulation environment developed in Chapter 3. However, the background traffic model and router drop characteristics are modified to evaluate protocol performance over a range of network drop rates and patterns. Key aspects of the simulation model are given below:

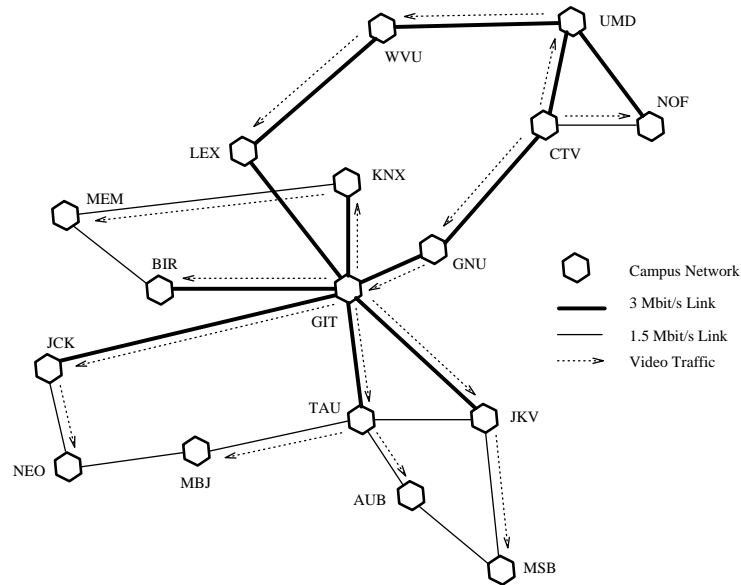


Figure 5.4: Wide-area network topology used in simulation experiments.

**Topology:** Figure 5.4 shows the network topology. It is closely based on the SURAnet backbone, a contemporary WAN interconnecting research institutions in the South-eastern US.

**Application traffic:** Application traffic is a 500-second trace of MPEG-I encoded video at 16 frames/second (fps), 320-by-240 pixel resolution and 8 bit color. The video stream bandwidth averages 220 Kbits/s and a peak rate of 530 Kbits/s.

**Routing:** The dotted arrows in Figure 5.4 show the multicast routing tree constructed for the video distribution from a source on the CTX campus. The multicast tree is based on shortest hop unicast routing.

**Packet drop rates:** The background traffic generated by each campus network is fixed at moderate load (i.e., 1mbps.) Packets are dropped at a router if there is insufficient buffer space at the output port. In addition to the standard tail-drop model, we control router drop patterns as follows: Upon packet arrival, a geometrically

distributed burst of packets with mean  $\mu$  will be dropped with probability  $\rho$  if the output queue is less than two-thirds full, and by  $10 \times \rho$  if the queue is greater than two-thirds full. Thus, routers drop packets in blocks for larger values of  $\mu$ , and, isolated drops for small values of  $\mu$ . The  $(\mu, \rho)$  model evaluates the performance of FEC and MESH-M under various drop patterns; that is, correlated block losses, versus isolated losses.

**Transmission delays:** The end-to-end packet delays in the network are composed of router queuing and the per-link transmission delay of the packet. Queueing delay results from congestion due to application and background traffic.

**Receiver operation:** As a conservative default value, receivers use a one second slack time for jitter and S-ARQ retransmission. Once the slack buffer is initially filled, complete video frames are removed at the 16 fps rate.

The simulation is configured with receivers located on all 17 campus networks each with an associated AR for the subgroup. Only one multicast source is assumed, and it is located at the CTV campus. In the experiments, ARs send advertisement messages once per second resulting in an advertisement traffic load of about 3% of the multicast video stream. In the base case, the error intensity values are set to  $\rho = 0.006$  and  $\mu = 3$ . These loss rates result in individual wide-area links losing between 1% and 6% of packets. These error rates are high enough to exercise MESH-M and are within the range of measured loss rates in the Internet.

### 5.3.2 MESH-M Retransmission Behavior

Figure 5.5 and Figure 5.6 present data from simulation experiments on the dynamic behavior of the MESH-M retransmission scheme. Figure 5.5 shows how MESH-M distributes retransmission requests across the set of ARs. Since the retransmission algorithm given in Figure 5.3 biases towards nearby ARs and the GIT campus connects to several campuses,

### 5.3. MESH-M Performance Evaluation 127

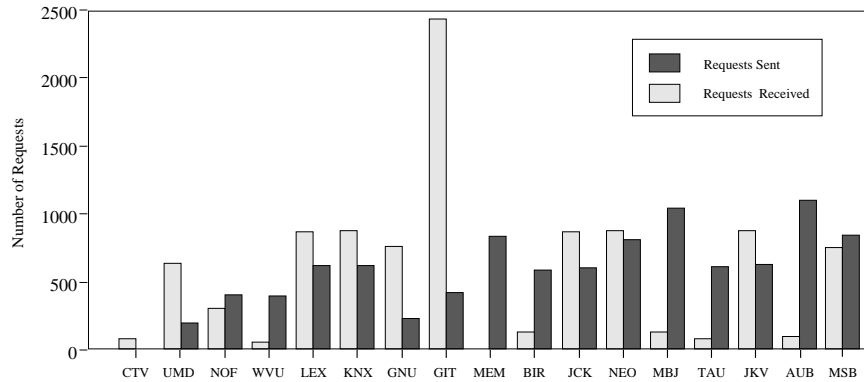


Figure 5.5: Distribution of retransmission requests.

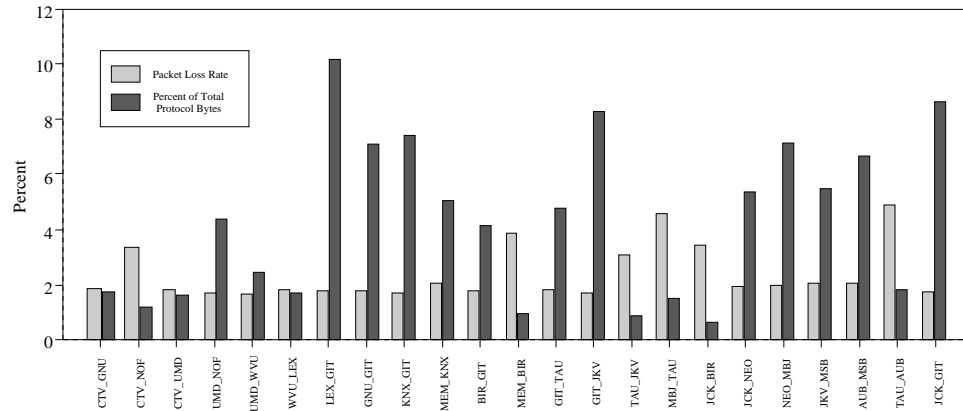


Figure 5.6: Retransmission protocol overhead distribution and drop rates per link.

the GIT AR receives a disproportionate number of requests. The absolute rate of retransmission requests arriving at the GIT AR is nonetheless modest for the scenario studied, i.e., less than 5 requests per second on average. In general, the effect shows that campuses with large fan-out in the wide-area topology may result in a concentration of protocol traffic at a single AR, but the threat of processing bottlenecks is small due to limited fan-out in WAN topologies and the processing capacities of modern workstations.

Figure 5.6 shows the distribution of MESH-M network overhead versus congestion in the network. Recall that MESH-M protocol traffic includes advertisements, retransmission

requests and retransmitted data packets. The data illustrates the tendency of the retransmission algorithm to distribute retransmission overhead, when possible, along network links with the least congestion, as indicated here by the packet loss rate on each link.

### 5.3.3 Comparison of MESH-M and Source-Based FEC

In this section we compare the performance of MESH-M to source-based Reed-Solomon FEC with overcoding levels ranging from 7.5–40%. Reed-Solomon codes add  $h$  redundant packets to  $n$  data packets [13]. If any  $n$  of the  $n + h$  packets are received, then the original data can be recovered. In our simulation experiments,  $H$  redundant packets are added to each group of packets from  $N$  frames, denoted here as a FEC ( $N:H$ ) encoding. Since each frame in the video traffic trace is composed of, on average, slightly more than two data packets, FEC (2:1) yields approximately 20% overcoding.

#### 5.3.3.1 Application Performance

Figure 5.7 shows the percentage of frames played at each campus for six different protocol scenarios: no error recovery, MESH-M, and FEC overcoding at 10% (5:1), 15% (3:1), 21% (4:2) and 29% (4:3). In these measurements, all the packets making up a video frame must be available at the playback deadline in order for the application to play a frame. Over all campuses, FEC (5:1) and FEC (3:1) improve application performance by 30% over the no-protocol case. The FEC (4:2) and (4:3) schemes improve the application performance by 49% and 61%, respectively. MESH-M provides the best error coverage, increasing application performance approximately 80%.

In Figure 5.7 the receivers are ordered left-to-right by their distance from the video source at campus CTV. Due to the cumulative effects of packet loss at each wide-area link, receivers farthest from the multicast source experience the highest loss rates and therefore the greatest performance gains from the use of error recovery techniques. As the distance

### 5.3. MESH-M Performance Evaluation 129

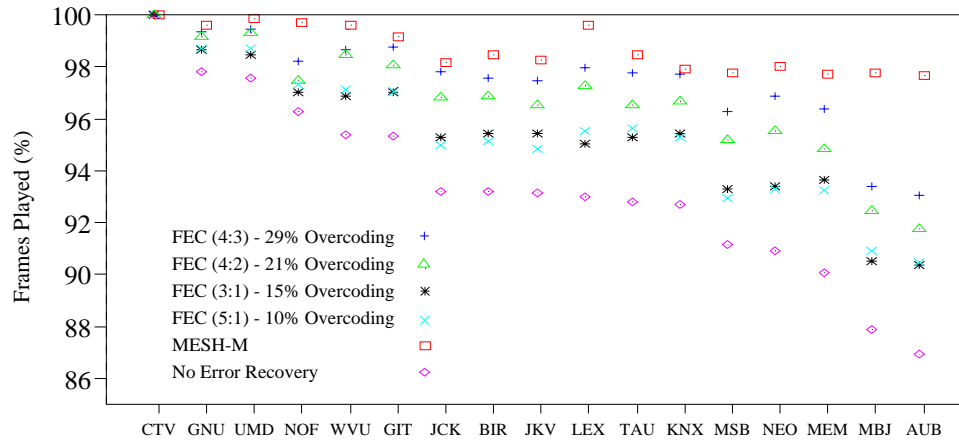


Figure 5.7: Application performance under FEC and MESH-M error control.

from the source increases, local retransmissions using MESH-M maintains the playback quality increasingly better than the source-based FEC approach.

#### 5.3.3.2 Network Cost

To compare the network overhead associated with error control schemes, a network cost metric is defined here. For each link in the WAN topology, the *link cost* is defined as the product of two factors: (1) the congestion of the link, as measured by its average packet loss rate over the duration of the data delivery, and (2) the amount of protocol traffic carried on the link, as measured by the ratio of protocol bytes to total traffic bytes on the link. The *network cost* is then the sum over all link costs. This definition reflects the importance of steering protocol overhead away from congested links.

Figure 5.8 plots the relative network cost for FEC overcoding levels ranging from 7.5–40% and MESH-M. The network cost of FEC increases linearly up to 35% overcoding and more rapidly thereafter due to the added congestion. The dotted line represents the cost metric calculated for the MESH-M retransmission scheme – which equals that of 7.5% overcoding. MESH-M actually adds 10% more traffic to the network than 7.5% FEC overcoding,

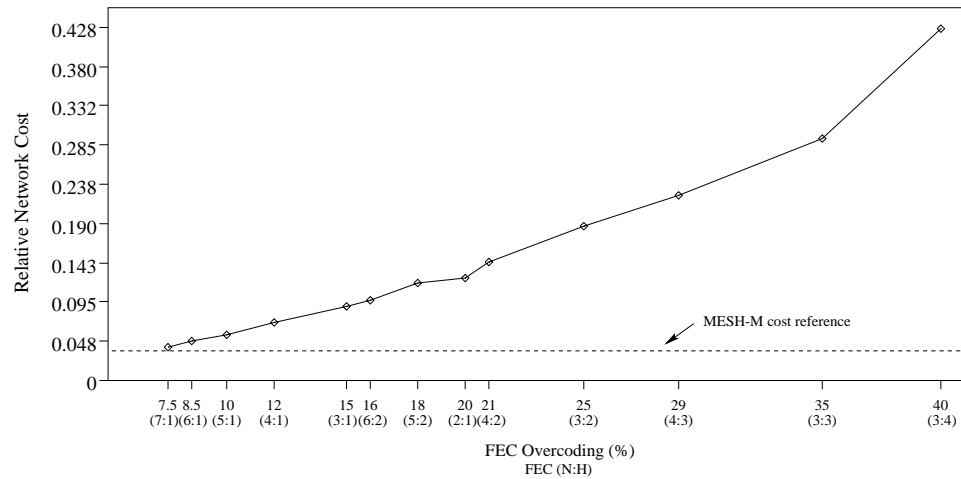


Figure 5.8: Network cost for FEC overcoding.

but MESH-M biases its traffic towards lightly loaded network links whereas FEC overhead is added on the links in the multicast routing tree.

### 5.3.3.3 Different Network Error Rates

The performance and network cost of FEC and MESH-M over different error intensities and burst loss sizes is compared here using FEC(5:1) 10% overcoding and FEC(4:2) 21% overcoding. The choice of FEC(5:1) was motivated by the similar cost of MESH-M (as shown in the previous section) while FEC(4:2) provides a comparison for a much higher amount of overcoding.

Recall that  $\rho$  and  $\mu$  characterize the loss/load error model at each network router. In Figures 9–12, the application performance and relative network cost of the MESH-M scheme along with FEC(5:1) and FEC(4:2) are plotted across a range of values for  $\rho$  and either  $\mu = 1$  or  $\mu = 3$ . All other simulation parameters remain as in the base configuration.

The error recovery performance and network cost of FEC(5:1), FEC(4:2), and MESH-M are shown for the isolated loss model, i.e.,  $\mu = 1$ , in Figure 5.9 and Figure 5.10. In this

### 5.3. MESH-M Performance Evaluation 131

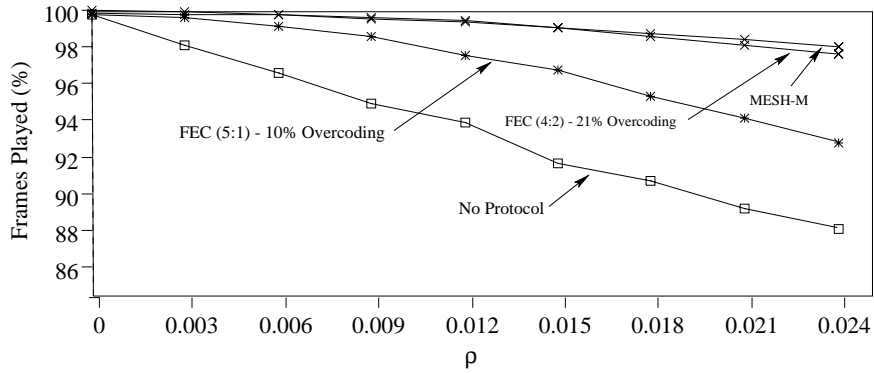


Figure 5.9: Application performance for FEC(4:2), FEC(5:1), and MESH-M with  $\mu = 1$ .

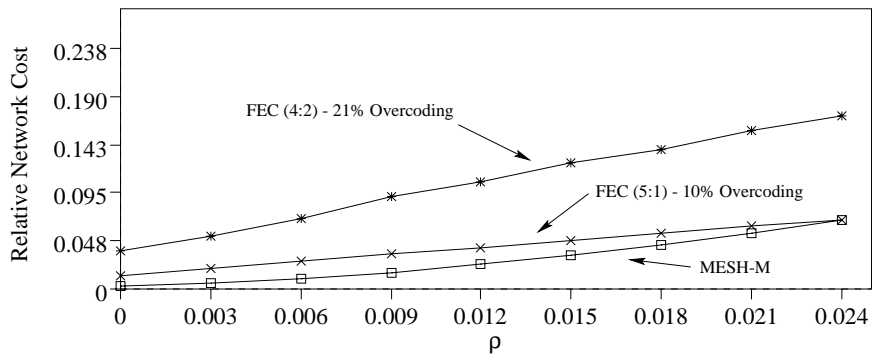


Figure 5.10: Network cost for FEC(4:2), FEC(5:1), and MESH-M with  $\mu = 1$ .

experiment, without protocol overhead, the link drop rates are between 0–2.7% for  $\rho = 0$ , and between 2.4–4.8% for  $\rho = 0.024$ .

As seen in Figure 5.9, the performance of FEC(4:2) and retransmission are comparable, and both techniques provide excellent error recovery over the full range of  $\rho$ . Figure 5.10 shows, however, that the cost metric for FEC(4:2) is roughly 2.5 to 16 times greater than MESH-M over the range of  $\rho$  shown. On the other hand, FEC(5:1) has a network cost similar to MESH-M, but its error recovery effectiveness is noticeably less, especially under higher error rates.



### 5.3. MESH-M Performance Evaluation 132

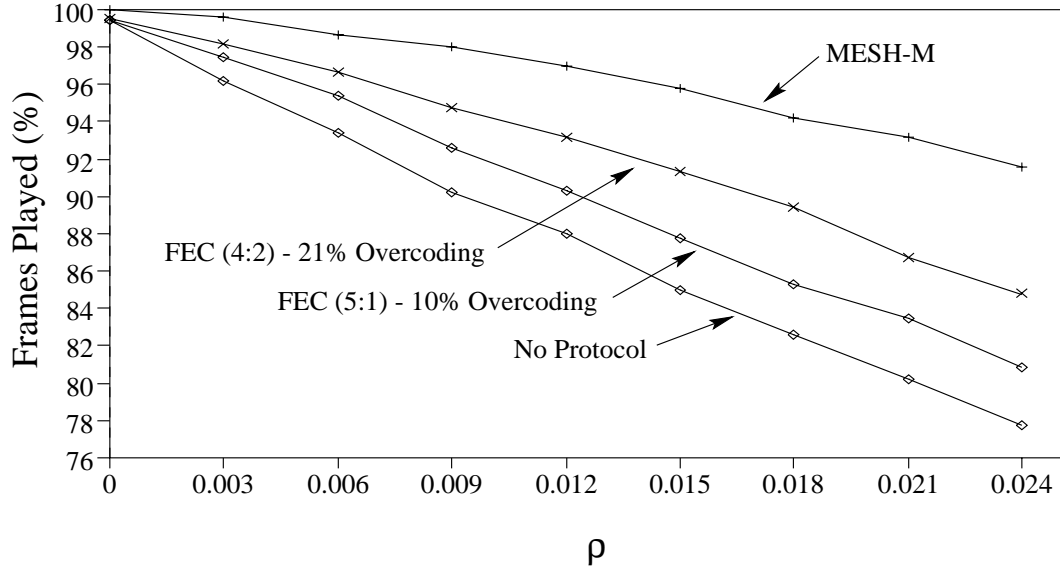


Figure 5.11: Application performance for FEC(4:2), FEC(5:1), and MESH-M with  $\mu = 3$ .

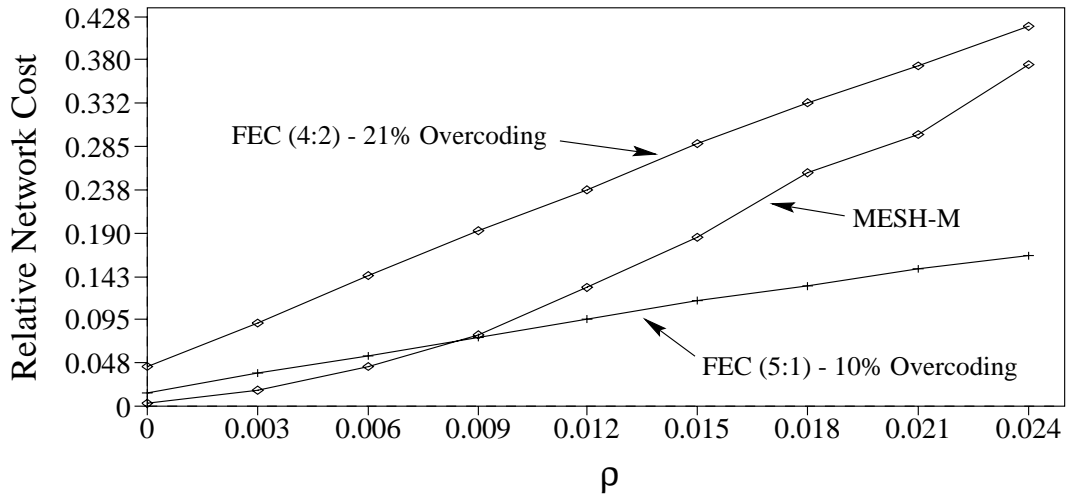


Figure 5.12: Network cost for FEC(4:2), FEC(5:1), and MESH-M with  $\mu = 3$ .

In Figure 5.11, the application performance of FEC(4:2), FEC(5:1) and MESH-M for bursty losses, i.e.,  $\mu = 3$ , is shown. For this network scenario, without protocol traffic, the link drop rates are between 0–2.8% for  $\rho = 0$  and between 6.9–10.1% for  $\rho = 0.024$ .

MESH-M performs significantly better than FEC(4:2) in this scenario, as seen in Figure 5.11. Figure 5.12 reveals that the network cost for MESH-M at low error rates is small, but it increases rapidly as error rates rise, due to the greater amount of retransmitted data. As a result, the cost for retransmission is similar to 10% overcoding at  $\rho = 0.009$ , but closer to the cost for 21% overcoding at  $\rho = 0.024$ . By contrast, the FEC schemes have penalties that grow linearly with  $\rho$ .

## 5.4 Conclusions and Future Work

This chapter presented MESH-M – a low-cost, effective error control for multimedia applications in wide-area multicast networks. Simulation experiments in this study give evidence that MESH-M succeeds in localizing retransmissions within the wide-area topology, and in steering retransmission traffic away from congested links. These properties enable MESH-M to scale to large receiver sets and heterogeneous networks. The experiments also show that, for applications which can tolerate some additional buffering delay, MESH-M provides excellent error control at a lower network overhead than end-to-end forward error correction.

While these performance and cost advantages come at the expense of protocol complexity within endsystems, MESH-M inherently requires less coordination overhead than schemes that set up and maintain per-source control trees, e.g., [47, 62, 103]. Furthermore, as compared with these other schemes, MESH-M is more robust to active receiver failures, adapts more readily to network congestion, and is well-suited for multiple sources. Our future work will extend MESH-M by incorporating FEC with distributed retransmissions, and combine MESH-R for full reliability of receiver feedback and other state critical to the multimedia application.

## Conclusions and Future Work

---

### 6.1 Conclusions

The rapid growth in networked computer systems and available bandwidth is driving interest in large-scale multicast applications such as distributed computing, multimedia, gaming, and bulk-data transfer. Wide-area multicast networks, however, will likely provide service comparable to that found in the Internet today; i.e., no performance guarantees on transmission delay, drop rate, or transmission delay variation. As a result, multicast transport services, such as error control and receiver feedback, are required by many multicast applications.

Central to transport protocol design is the control structure which distributes protocol processing, localizes error repair, and aggregates receiver state. Further, the control structure determines the flow of end-system state used by feedback-based algorithms, e.g., congestion and error control protocols at the source. Thus, the design of control structure is critical since it determines the services that can be provided at the transport layer, as well as its scalability, efficiency, and performance.

This dissertation presented a novel, fully-distributed transport control framework called MESH. Our key contributions include:

- Domain-based control structure: MESH partitions the multicast group into an hierarchical control structure based on natural network domain boundaries found in networks today (e.g., local, campus, and backbone domains).
- State synchronization: MESH uses a data-driven, multicast feedback scheme to synchronize group state within each domain. Special group members, called *active receivers* (ARs), aggregate and propagate state throughout the domain hierarchy.
- Performance-based error control: Sophisticated heuristics localize the recovery of messages dropped by the network between group members. The heuristics are driven by dynamic network performance characteristics such as delay and error patterns.

### 6.1.1 Large-Scale Reliable Transport: MESH-R

Using the MESH framework, we designed a reliable transport (MESH-R) suitable for use by a large-scale, bulk-data distribution application. Key contributions of MESH-R include (1) a robust state synchronization protocol that provides detailed end-system state for reliability, congestion control, group management, and other end-system services, and (2) a receiver-driven error control service which uses a self-organizing, soft-state unicast recovery structure between multicast receivers.

Using a high-fidelity network simulation of SURAnet and vBNSnet, we validated correct operation of MESH-R under a range of network loads. The study demonstrated that MESH's state synchronization provides low-latency, efficient state feedback to the source. The study demonstrated that MESH-R's error control protocol adapts rapidly to changing network conditions, exhibits good locality in recovering lost packets, and effectively exploits redundant cross-links outside of the multicast routing tree.

We then compared the relative performance and network overhead of MESH-R to other classes of solutions found in the literature (a centralized, a tree-based, and an unstructured protocol). The experiments showed that (1) MESH-R achieves the lowest file distribution

latency in both SURAnet and vBNSnet, (2) MESH-R offers the best network efficiency in the high-speed vBNSnet environment, and (3) MESH-R offers network efficiency comparable to an optimal tree-based approach in SURAnet.

### 6.1.2 Large-Scale Deadline-Driven Transport: MESH-M

We considered the design of transport service for large-scale, continuous media applications. In the continuous-media domain, transport service requirements include network performance feedback and timely recovery of messages lost by the network (since each message has an associated play-back deadline.)

We developed MESH-M, a distributed error control scheme which recovers dropped messages using the time-constrained retransmission model of S-ARQ. The key contribution of MESH-M is the error control framework. Our approach is based on identifying local nodes which have a high probability of retransmitting messages within the media's playback deadline. Our approach attempts to minimize the network overhead by avoiding congested network links and localize repair. Unlike other schemes, our approach is implemented entirely at the transport layer, relies on a run-time algorithm for dynamic configuration of the control framework, and performs retransmission-based recovery based upon observed network delays and error patterns relative to the source.

We compared the performance of MESH-M to a range of FEC overcoding levels. Our analysis focused on a single-source, MPEG video-cast application in the SURAnet environment. Key results of the analysis include:

- MESH-M effectively distributed error recovery throughout the group.
- MESH-M distributed retransmission overhead, when possible, along network links with the least congestion.
- MESH-M improved application performance by 80% – a 30% improvement over FEC schemes that overcode the stream by 29%.

- MESH-M is very efficient – comparable to FEC overcoding at 7.5%, and more than three times more efficient than a 29% FEC overcoding scheme.

## 6.2 Future Work

In this dissertation, we showed (for the network environments considered) that the generic implementation of MESH-R was very attractive for reliable multicast, and MESH-M performed very well for deadline-driven reliable multicast. In general, however, no two multicast sessions are the same. Each application has different error control and feedback requirements. Likewise, each network has its own particular performance characteristics. One key feature of the MESH framework is that the heuristics can be programmed for a particular application, network environment, or group size. Future research directions focus on developing MESH protocols for other application domains, and lowering the network overhead associated with MESH-R and MESH-M without compromising application performance. These directions are summarized below.

- In MESH, each node domain-multicasts state according to the application’s requirements. Domain-multicast has a number of advantages in that it is robust and provides low-latency synchronization. However, it is an expensive operation. In future work, we will investigate techniques that lower the state frequency by developing an API which allows the application or group management protocol to identify state critical to the progress of the group. We will also investigate techniques which can efficiently exchange state within a domain, without sacrificing synchronization latency. The hypercube technique proposed by Liebeherr [61] is a promising approach.
- MESH’s error recovery heuristics do very well localizing error recovery simply by considering network delay and loss characteristics. In future work, we will consider other heuristics that might improve recovery latency and lower network overhead. In particular, we believe that adapting the retransmission timers to recovery delay and

duplicates can greatly improve MESH's efficiency. Also, we will investigate modifying the server selection algorithm to consider past recovery behavior and loss rates between ARs.

- The MESH framework inherently supports multi-source applications. However, multiple sources increase the size and frequency of state messages. In future work, we will investigate low-overhead state services for multisource applications. Of particular interest is a web-cache update application. Here, once a server receives an update, it must quickly inform the other servers which data are no longer valid. Updating the stale data is a separate issue in which multicast may not be the optimal technique. We plan to investigate low latency, low overhead web-cache solutions using MESH.
- MESH relies on a retransmission-only error control scheme. However, FEC has been shown to recover a large number of network losses without retransmission. In future work, we will investigate integrating FEC with MESH-based retransmission. We believe a hybrid scheme can significantly lower the network overhead and service latency.
- The MESH framework hinges on the ability to localize multicast messages within network domains. Since IPv6 is not widely deployed, it is unclear if domain-multicast can be relied upon. Although centralized approaches and local multicast addressing present viable options, we plan to investigate the feasibility of a multicast gateway service located at each network exchange point. In addition to domain-scoping service, the gateway can provide a number of critical services such as congestion control, accounting, metering, and quality of service monitoring.

## Self-Similar Traffic

---

### A.1 Generating Self-Similar Traffic Using the FFT Method

The FFT strategy to generating FGN is based on constructing a sequence of complex numbers  $\{z_0, \dots, z_{n-1}\}$  corresponding to the power spectrum  $f(\lambda; H)$  of a FGN process. Applying the inverse discrete fourier transform to  $\{z\}$  obtains a sequence  $\{x_0, \dots, x_{n-1}\}$  corresponding to the time domain counterpart to  $\{z\}$ . Since  $\{x\}$  has by construction the power spectrum of FGN, the sample path is guaranteed to have the autocorrelation properties of an FGN process. The steps to the FFT technique are given below [79]:

1. Construct a sequence of values  $f_1, \dots, f_n$ , where  $f_i = f(2\pi i/n; H)$ . The power spectrum of FGN, developed by Whittle, is given below by Beran [12]:

$$f(\lambda; H) = \mathcal{A}(\lambda; H)[|\lambda|^{-2H-1} + \mathcal{B}(\lambda; H)] \quad (\text{A.1})$$

$$\mathcal{A}(\lambda; H) = 2 \sin(\pi H), (2H + 1)(1 - \cos \lambda) \quad (\text{A.2})$$

$$\mathcal{B}(\lambda; H) = \sum_{j=1}^{\infty} [(2\pi j + \lambda)^{-2H-1} + (2\pi j - \lambda)^{-2H-1}] \quad (\text{A.3})$$

2. Multiply each  $f_i$  by an independent exponential random variable with mean 1, creating a sequence  $\{\hat{f}_i\}$ . This step distributes the power for a given frequency as an independent



### A.1. Generating Self-Similar Traffic Using the FFT Method 140

exponential random variable such that the mean is equal to the actual power. This step is necessary because estimating the power spectrum using the Fourier transform relies on the fact that ordinates (called the periodogram ordinates) are asymptotically independent and exponentially distributed with mean  $f(\lambda; H)$ .

3. Construct a sequence of complex numbers  $\{z_0, \dots, z_{n-1}\}$ , where  $|z_i| = \sqrt{\hat{f}_{i-1}}$  and the phase of  $z_i$  is uniformly distributed between 0 and  $2\pi$ . The random phase technique, given by Schiff [89], maintains the autocorrelation structure of the FGN process while ensuring that each sample path is unique.
4. Construct  $\{z'_0, \dots, z'_{n-1}\}$  as follows:

$$z'_j = \begin{cases} 0, & \text{if } j = 0 \\ z_j, & \text{if } 0 < j \leq n/2 \\ \overline{z_{n-j}}, & \text{if } n/2 < j < n \end{cases} \quad (\text{A.4})$$

where  $\overline{z_{n-j}}$  is the complex conjugate of  $z_{n-j}$ . Since  $\{z'_j\}$  is symmetric about  $z'_{n/2}$ , it corresponds to the Fourier transform of a real-valued signal [89].

5. Take the inverse-Fourier transform of  $\{z'_j\}$ , resulting in an approximate FGN sample path  $\{x_0, \dots, x_{n-1}\}$ .

# B

## Simulated Network Delay and Error Rates

---

### B.1 Simulated Round-trip Delays and Drop Rates in SURAnet

	uva	umd	nof	wvu	lex	knx	gnu	git	mem	bir	jck	neo	mbj	tau	jkv	aub	msb
uva	0	21	21	26	32	31	23	28	38	33	37	40	38	33	33	37	40
umd	21	0	20	23	28	35	28	32	43	37	41	44	42	37	37	41	43
nof	21	21	0	26	32	35	27	31	42	36	40	44	42	36	36	41	43
wvu	26	23	26	0	23	32	33	29	39	34	37	41	39	34	34	38	40
lex	32	28	31	23	0	26	27	23	34	29	32	36	34	29	29	33	34
knx	31	34	34	31	26	0	25	21	25	26	30	34	32	26	26	30	32
gnu	23	27	27	32	27	24	0	21	32	27	31	34	33	27	27	31	33
git	27	31	31	28	23	20	21	0	28	23	27	31	29	23	23	27	29
mem	38	42	42	38	34	26	33	28	0	21	25	29	32	33	33	38	40
bir	33	37	37	33	28	26	27	23	21	0	21	25	29	29	29	33	34
jck	37	41	41	38	33	31	32	28	25	21	0	21	25	33	33	37	39
neo	41	45	45	41	36	34	35	31	29	25	21	0	21	27	31	31	36
mbj	39	43	43	39	34	32	33	29	32	29	25	21	0	24	27	27	32
tau	33	37	37	33	29	26	27	23	34	29	32	27	23	0	21	21	26
jkv	33	37	37	33	29	26	27	23	34	29	32	31	27	21	0	26	23
aub	37	41	41	37	32	30	31	27	38	32	36	31	27	21	26	0	21
msb	39	43	43	39	34	33	33	29	40	34	38	35	32	26	23	21	0

Table B.1: Suranet round-trip network delay (in *ms*) for  $\overline{M} = 10$  packets per  $100ms$ .

### B.1. Simulated Round-trip Delays and Drop Rates in SURAnet 142

	uva	umd	nof	wvu	lex	knx	gnu	git	mem	bir	jck	neo	mbj	tau	jkv	aub	msb
uva	0	363	22	370	376	608	361	605	620	610	615	621	618	611	611	617	618
umd	363	0	21	24	31	956	709	952	968	959	962	967	966	960	958	963	965
nof	22	21	0	27	34	612	365	609	624	615	622	626	623	617	617	621	624
wvu	370	24	27	0	24	34	716	30	45	36	40	44	43	36	36	42	44
lex	377	30	33	23	0	28	267	24	38	30	33	38	37	30	30	36	37
knx	611	957	615	33	27	0	265	21	26	28	31	36	35	28	28	33	35
gnu	359	708	364	716	266	264	0	261	273	266	269	274	272	265	267	272	275
git	606	953	611	29	23	21	261	0	32	24	27	33	31	24	24	29	31
mem	620	967	626	44	38	26	276	32	0	24	27	32	35	38	38	44	45
bir	611	959	617	36	30	28	267	24	23	0	21	26	30	30	30	36	37
jck	617	964	622	40	34	32	273	28	27	21	0	22	26	34	34	40	42
neo	622	970	628	45	39	37	277	33	32	26	22	0	21	28	32	33	38
mbj	620	968	627	44	37	35	276	31	35	30	26	21	0	25	28	29	34
tau	611	960	618	36	30	28	267	24	39	31	34	28	24	0	21	22	27
jkv	612	959	617	36	30	28	268	24	39	30	34	32	28	21	0	27	24
aub	618	965	624	42	35	33	274	29	45	35	39	32	29	22	27	0	21
msb	620	966	626	43	37	35	276	31	46	37	41	37	34	27	25	21	0

Table B.2: Suranet round-trip network delay (in *ms*) for  $\bar{M} = 70$  packets per 100*ms*.

	uva	umd	nof	wvu	lex	knx	gnu	git	mem	bir	jck	neo	mbj	tau	jkv	aub	msb
uva	0	3	2	4	6	6	2	4	6	5	5	7	6	5	6	6	6
umd	3	0	3	3	4	7	4	5	7	7	7	7	7	7	6	8	8
nof	3	3	0	4	5	7	4	5	8	7	6	8	7	7	7	8	7
wvu	4	3	4	0	3	5	6	4	6	6	5	7	6	5	5	6	6
lex	6	4	6	3	0	4	4	3	5	4	4	5	5	4	4	5	5
knx	5	7	7	5	4	0	4	3	3	4	4	5	5	4	4	6	5
gnu	3	4	4	6	4	4	0	2	4	4	4	5	5	4	4	6	5
git	4	6	5	4	3	3	3	0	4	3	3	4	4	3	3	4	4
mem	7	8	8	7	6	3	6	4	0	2	4	5	7	6	6	6	6
bir	5	7	6	5	4	4	4	3	2	0	3	4	5	4	4	5	5
jck	5	7	7	5	4	4	4	3	4	3	0	3	4	4	4	5	5
neo	7	8	8	6	6	5	5	4	5	4	3	0	3	4	5	5	6
mbj	7	7	7	6	5	6	5	4	7	5	4	3	0	3	4	4	5
tau	5	7	7	5	4	4	4	2	5	5	4	5	3	0	2	3	4
jkv	5	7	6	5	4	4	4	3	6	4	4	5	4	2	0	4	2
aub	7	7	8	7	6	6	6	4	7	5	5	5	4	3	4	0	3
msb	7	7	8	6	5	5	5	4	7	5	5	6	5	4	3	2	0

Table B.3: Suranet round-trip network drop percent for  $\bar{M} = 10$  packets per 100*ms*.

## B.2. Simulated Round-trip Delays and Drop Rates in vBNSnet 143

	uva	umd	nof	wvu	lex	knx	gnu	git	mem	bir	jck	neo	mbj	tau	jkv	aub	msb
uva	0	14	3	16	18	31	9	29	31	31	31	33	32	30	30	32	31
umd	14	0	3	3	4	39	24	39	42	41	40	40	42	42	41	40	39
nof	3	3	0	4	5	32	11	30	31	33	33	33	33	31	32	33	34
wvu	16	3	4	0	3	5	25	3	7	5	5	7	6	5	6	6	6
lex	18	4	5	3	0	4	24	3	6	4	4	5	5	4	4	5	5
knx	30	40	33	5	4	0	23	3	3	3	4	5	6	4	4	5	5
gnu	10	21	11	25	22	23	0	21	24	21	23	25	25	22	22	24	26
git	27	38	32	4	3	3	20	0	4	3	3	3	4	3	3	4	4
mem	34	43	38	7	5	3	30	4	0	2	4	5	7	6	5	7	6
bir	31	40	35	5	4	4	25	3	3	0	3	4	5	4	4	6	5
jck	33	40	37	4	4	4	28	3	4	3	0	3	4	4	4	5	6
neo	36	43	39	6	5	5	30	4	5	4	3	0	3	4	5	6	6
mbj	36	43	38	6	5	5	30	4	6	5	4	3	0	3	4	4	5
tau	31	40	35	5	4	4	26	2	5	4	4	5	3	0	3	2	4
jkv	32	40	35	5	4	4	25	3	5	4	4	5	4	3	0	4	3
aub	35	43	37	7	5	5	29	4	6	5	5	6	4	3	4	0	3
msb	36	43	38	6	5	5	30	4	7	5	5	7	5	4	3	3	0

Table B.4: Suranet round-trip network drop percent for  $\bar{M} = 70$  packets per 100ms.

## B.2 Simulated Round-trip Delays and Drop Rates in vBN-Snet

	sfc	ncsr	sdsc	den	chi	ncsa	hou	cle	cor	psc	mae
sfc	0	29	24	26	35	42	39	43	47	53	53
ncsr	28	0	35	20	29	33	44	37	42	42	51
sdsc	24	35	0	33	42	39	32	50	55	53	46
den	26	20	32	0	26	30	42	35	39	39	49
chi	34	28	42	26	0	22	28	26	30	31	40
ncsa	42	33	38	31	22	0	24	31	35	35	38
hou	38	45	32	42	29	24	0	37	42	39	32
cle	43	37	51	35	26	31	37	0	22	22	28
cor	48	42	55	39	31	35	42	22	0	27	33
psc	54	42	53	39	31	35	39	22	26	0	24
mae	53	51	46	48	40	39	32	29	33	24	0

Table B.5: vBNSnet round-trip delays (in ms) for  $\bar{M} = 700$  packets per 100ms.

## B.2. Simulated Round-trip Delays and Drop Rates in vBNSnet 144

	sfc	ncsr	sdsc	den	chi	ncsa	hou	cle	cor	psc	mae
sfc	0	35	29	28	51	61	49	79	86	86	68
ncsr	35	0	46	24	46	63	67	75	81	83	98
sdsc	29	46	0	40	67	49	37	95	102	65	56
den	28	24	40	0	40	57	60	69	76	78	93
chi	51	46	67	40	0	35	47	47	54	56	72
ncsa	61	63	49	57	34	0	30	63	70	73	49
hou	49	66	37	60	47	30	0	76	82	46	37
cle	79	74	95	69	47	64	75	0	25	28	37
cor	87	82	103	77	55	71	82	25	0	36	44
psc	86	83	66	79	57	73	46	28	36	0	27
mae	68	99	57	94	72	49	37	36	44	27	0

Table B.6: vBNSnet round-trip delays (in ms) for  $\overline{M} = 2700$  packets per 100ms.

	sfc	ncsr	sdsc	den	chi	ncsa	hou	cle	cor	psc	mae
sfc	0	2	2	2	2	3	3	3	3	3	3
ncsr	2	0	3	2	3	3	3	3	3	4	4
sdsc	2	3	0	3	3	2	2	3	4	3	2
den	2	2	2	0	2	2	3	2	3	3	3
chi	2	2	3	2	0	2	2	2	2	2	3
ncsa	3	3	3	3	2	0	2	3	3	3	2
hou	3	3	2	3	2	2	0	3	3	3	2
cle	3	3	4	2	2	2	2	0	2	2	2
cor	3	3	4	3	2	3	3	2	0	2	3
psc	4	4	3	3	2	3	2	2	2	0	2
mae	3	4	3	3	3	2	2	2	3	2	0

Table B.7: vBNSnet round-trip drop percent for  $\overline{M} = 700$  packets per 100ms.

	sfc	ncsr	sdsc	den	chi	ncsa	hou	cle	cor	psc	mae
sfc	0	4	5	3	17	10	7	34	35	23	11
ncsr	4	0	7	3	18	20	11	33	35	35	33
sdsc	5	7	0	5	21	7	5	38	37	9	8
den	3	4	5	0	15	21	11	31	33	33	31
chi	16	16	21	14	0	7	10	22	24	23	22
ncsa	10	21	8	20	8	0	4	29	25	28	9
hou	7	12	5	11	10	4	0	30	31	7	6
cle	34	34	37	31	22	26	29	0	2	4	5
cor	34	33	35	32	23	29	30	3	0	4	7
psc	22	34	9	34	24	27	6	4	5	0	2
mae	12	33	8	32	24	8	5	5	6	3	0

Table B.8: vBNSnet round-trip drop percent for  $\overline{M} = 2700$  packets per 100ms.

## Bibliography

---

- [1] Snoop Packet Filter. Sun Solaris 2.5 man page. Sun Microsystems, 1996.
- [2] ATM Forum, ATM User-Network Interface Specification Version 3.0, 1993.
- [3] SURAnet Backbone Network Architecture, October 1994. Available by ftp at ftp.sura.net in pub/maps/SURAnet/SURA.backbone.3.ps.
- [4] Xpress Transport Protocol Specification, Version 4.0, 1994. XTP Forum, Available at <http://www.ca.sandia.gov/xtp/xtp.html>.
- [5] ATM Forum, ATM Forum Traffic Management Specification Version 4.0, Contribution 95-0013R11, March 1996.
- [6] NSF very High Speed Backbone Network Service Management and Operations Monthly Report, MCI vBNS Engineering, April 1997.
- [7] A. Adas and A. Mukherjee. On Resource Management and QoS Guarantees for Long Range Dependent Traffic. In *Proc. IEEE Infocom*, pages 779–787, April 1995.
- [8] J. S. Ahn and P. B. Danzig. Packet Network Simulation: Speedup and Accuracy Versus Timing Granularity. *IEEE/ACM Transactions on Networking*, 4:743–757, 1996.

- [9] M. Ammar and L. Wu. Improving the Performance of Point to Multi-Point ARQ Protocols through Destination Set Splitting. *IEEE INFOCOM '92*, pages 262–271, May 1992.
- [10] S. Armstrong, A. Freier, and K. Marzullo. Multicast Transport Protocol. Technical Report RFC 1301, Internet Engineering Task Force, February 1992.
- [11] D. G. Basset. Reliable Multicast Services for Tele-collaboration. Masters Thesis, University of Virginia, January 1997.
- [12] J. Beran. Statistical Methods for Data with Long-Range Dependence. In *Statistical Science*, 7(4), pages 404–427, 1992.
- [13] E. Biersack. Performance Evaluation of Forward Error Correction in ATM Networks. *ACM SIGCOMM '92*, 22(4):248–258, August 1992.
- [14] K. Birman. The Process Group Approach to Reliable Distributed Computing. *Communications of the ACM*, 36(12):37–53, December 1993.
- [15] K. Birman and T. A. Joseph. Reliable Communication in the Presence of Failures. *ACM Transactions on Computer Systems*, 5(1):47–76, February 1987.
- [16] K. Birman, A. Schiper, and P. Stephenson. Lightweight Causal and Atomic Group Multicast. *ACM Transactions on Computer Systems*, 9(3):272–314, August 1991.
- [17] J. Bolot, T. Turletti, and I. Wakeman. Scalable Feedback Control for Multicast Video Distribution in the Internet. *ACM SIGCOMM '94*, 24(4):58–67, September 1994.
- [18] C. Bormann, J. Ott, H. C. Gehrcke, T. Kerschhat, and N. Seifert. MTP-2: Towards Achieving the S.E.R.O. Properties for Multicast Transport. In *ICCCN '94*, San Francisco, California, September 1994.
- [19] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview, RFC 1633, July 1994.

- [20] R. Braudes and S. Zabele. Requirements for multicast protocols. Request for Comments (Informational) RFC 1458, Internet Engineering Task Force, May 1993.
- [21] J. Chang and N. F. Maxemchuk. Reliable Broadcast Protocols. *ACM Transactions on Computer Systems*, 2(3):251–273, August 1984.
- [22] D. R. Cheriton. VMTP: A Transport Protocol for the Next Generation of Communication Systems. In *Proc. Sigcomm '86*, pages 406–415, Stowe, Vermont, August 1986.
- [23] D. R. Cheriton and C. L. Williamson. VMTP as the Transport Layer for High Performance Distributed Systems. *IEEE Communications Magazine*, 27(6):37–44, June 1989.
- [24] D. R. Cheriton and W. Zwaenepoel. Distributed Process Groups in the V Kernel. *ACM Transactions on Computer Systems*, 3(2):77–107, May 1985.
- [25] S. Cheung and M. Ammar. Using Destination Set Grouping to Improve the Performance of Window-Controlled Multipoint Connections. Technical Report GIT-CC-94-32, Georgia Institute of Technology, August 1994.
- [26] F. Christian, H. Aghili, R. Strong, and D. Dolev. Atomic Broadcast: From Simple Message Diffusion to Byzantine Agreement. Technical Report IBM Research Report RJ 5244 (54244), IBM Almaden Research Center and Hebrew University, July 1986.
- [27] D.D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanisms. In *Proc. Sigcomm '92*, pages 14–26, August 1992.
- [28] Douglas E. Comer. *Internetworking with TCP/IP, Second Edition*. Prentice Hall, 1991.



- [29] D. R. Cox. Long-Range Dependence: A Review. In *Statistics: An Appraisal, Proc. 50th Anniversary Conference*, pages 55–74, Ames, IA: Iowa State University Press. Iowa State Univ. Press, 1984.
- [30] J. Crowcroft and K. Paliwoda. A Multicast Transport Protocol. In *Proc. Sigcomm '88*, pages 247–256, Stanford, California, August 1988. ACM.
- [31] P. B. Danzig, S. Jamin, R. Caceres, D. Mitzel, and D. Estrin. An Empirical Workload Model for Driving Wide-area TCP/IP Network Simulations. *Internetworking: Research and Experience*, 3(1):1–26, March 1992.
- [32] S. Deering. Host extensions for IP multicasting. Request for Comments (Standard) STD 5, RFC 1112, Internet Engineering Task Force, August 1989. (Obsoletes RFC0988).
- [33] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. An Architecture for Wide-Area Multicast Routing. *ACM SIGCOMM '94*, 24(4):126–135, September 1994.
- [34] S. E. Deering. Multicast Routing in Internetworks and Extended LANs. In *Proc. Sigcomm '88*, pages 55–64, Stanford, California, August 1988. ACM.
- [35] S. E. Deering. *Multicast routing in a datagram internetwork*. PhD thesis, Stanford University, Palo Alto, California, December 1991.
- [36] B. J. Dempsey. *Retransmission-Based Error Control for Continuous Media Traffic in Packet-Switched Networks*. PhD thesis, Department of Computer Science, University of Virginia, May 1994.
- [37] B. J. Dempsey, J. Liebeherr, and A. C. Weaver. On Retransmission-Based Error Control for Continuous Media Traffic in Packet-Switching Networks. *Computer Networks and ISDN Systems*, 28(5):719–736, March 1996.

- [38] B. J. Dempsey, M. T. Lucas, and A. C. Weaver. An Empirical Study of Packet Voice Distribution over a Campus-Wide Network. *19th IEEE Conference on Local Computer Networks*, October 1994.
- [39] B. J. Dempsey, M. T. Lucas, and A. C. Weaver. Design and Implementation of a High-Quality Video Distribution System using XTP Reliable Multicast. In *Multimedia: Advanced Teleservices and High-Speed Communication Architectures*, Ralf Steinmetz (Editor), *Spring-Verlag*, pages 376–387, Heidelberg, Germany, September 1994.
- [40] C. Diot, W. Dabbous, and J. Crowcroft. Multipoint Communications: A Survey of Protocols, Functions, and Mechanisms. *IEEE Journal on Selected Areas in Communications*, 15(3), April 1997.
- [41] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. In *ACM Sigcomm '95*, pages 342–356, Cambridge, Massachusetts, September 1995.
- [42] R. Frederick. Nv manual pages.
- [43] R. Frederick. Multicast Routing Algorithms: A Survey, January 1994. Unpublished Memorandum Available at <ftp://ftp.csc.ncsu.edu/pup/rtcomm/RTMulticast.ps>.
- [44] H. Garcia-Molina and A. Spauster. Ordered and Reliable Multicast Communication. *ACM Transactions on Computer Systems*, 9(3):242–271, August 1991.
- [45] M.W. Garrett and W. Willinger. Analysis, Modeling and Generation of Self-Similar VBR Video Traffic. In *ACM Sigcomm 1994*, London, UK, August 1994.
- [46] H. Hefes and D. M. Lucantoni. A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance. *IEEE Journal on Selected Areas in Communications*, SAC-4(6):856–868, September 1986.

- [47] H. W. Holbrook, S. K. Singhal, and D. R. Cheriton. Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation. In *Proc. Sigcomm '95*, pages 328–341, August 1995.
- [48] V. Jacobson. Congestion Avoidance and Control. *Computer Communication Review*, 18(4):314–329, August 1988.
- [49] V. Jacobson. Multimedia Conferencing on the Internet. In *Tutorial 4, ACM Sigcomm '94*, September 1994.
- [50] V. Jacobson and S. McCanne. The LBL audio tool vat. Manual page, July 1992.
- [51] R. Jain and S. A. Routhier. Packet Trains: Measurements and a New Model for Computer Network Traffic. *IEEE Journal on Selected Areas in Communications*, SAC-4(6):986–995, September 1986.
- [52] M. Jones, S-A. Sorenson, and S. Wilbur. Protocol Design for Large Group Multicasting: The Message Distribution Protocol. *Computer Communications*, 14(5):287–297, June 1991.
- [53] M. F. Kaashoek, A. S. Tanenbaum, S. F. Hummel, and H. E. Bal. An Efficient Reliable Broadcast Protocol. *Operating Systems Review*, 23(4):5–19, October 1989.
- [54] V. Kompella, P. Vachaspathi and J. C. Pasquale, and G. C. Polyzos. Two Techniques for Multicasting for Multimedia Applications. Technical report, University of California, San Diego, November 1991.
- [55] S. Kramer. *Total Ordering of Messages in Multicast Communication Systems*. PhD thesis, The Hebrew University of Jerusalem, Jerusalem, Israel, December 1992.
- [56] W. Lau, A. Erramilli, J. L. Wang, and W. Willinger. Self-Similar Traffic Parameter Estimation: A Semi-Parametric Periodogram-Based Algorithm. In *Proc. IEEE Globecom '95*, Singapore, 1995.

- [57] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the Self-Similar Nature of Ethernet Traffic (Extended Version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, February 1994.
- [58] B. Levine, D. Lavo, and J. J. Garcia-Luna-Aceves. The case for reliable concurrent multicasting using shared ack trees. In *Proceedings of ACM Multimedia '96*, November 1996.
- [59] X. Li, S. Paul, P. Panch, and M. Ammar. Layered Video Multicast with Retransmission (LVMR): Evaluation of Error Recovery Schemes. In *Seventh International Workshop on Network and Operating Systems Support for Digital Audiovisual (NOSSDAV '97)*, May 1997.
- [60] X. Li, S. Paul, P. Panch, and M. Ammar. Layered Video Multicast with Retransmission (LVMR): Evaluation of Heirarchical Rate Control. In *Proc. IEEE Infocom '98*, San Francisco, Ca., March 1998.
- [61] J. Liebeherr and B. S. Sethi. Optimum Routing of Multicast Streams. In *Proc. IEEE Infocom '98*, San Francisco, Ca., March 1998.
- [62] J. C. Lin and S. Paul. RMTP: A Reliable Multicast Transport Protocol. In *Proc. IEEE Infocom '96*, pages –, March 1996.
- [63] M. T. Lucas, B. J. Dempsey, and A. C. Weaver. Distributed Error Recovery for Continuous Media Data in Wide-Area Networks. Technical Report CS-95-52, University of Virginia, July 1995.
- [64] M. T. Lucas, B. J. Dempsey, and A. C. Weaver. An Efficient Self-Similar Traffic Model for Wide-Area Network Simulation. In *Proceedings of IEEE Globecom '97*, pages 1572–1576, November 1997.

- [65] M. T. Lucas, B. J. Dempsey, and A. C. Weaver. MESH: Distributed Error Recovery for Multimedia Streams in Wide-Area Multicast Networks. In *Proceedings of IEEE International Conference on Communication (ICC '97)*, pages 1127–1132, June 1997.
- [66] M. T. Lucas and D. E. Wrege and B. Dempsey and A. C. Weaver. Statistical Characterization of Wide-Area IP Traffic. In *Proceedings of Sixth International Conference on Computer Communications and Networks (IC3N'97)*, pages 442–447, September 1997.
- [67] S. Maffeis, W. Bischofberger, and K. Matzel. A Generic Multicast Transport Service to Support Disconnected Operation. In *Second USENIX Symposium on Mobile and Location-Independent Computing*, Ann Arbor, MI, April 1995.
- [68] S. McCanne, V. Jacobsen, and M. Vetterli. Receiver-driven Layered Multicast. In *ACM Sigcomm '96*, pages 117–130, Stanford University, California, October 1996.
- [69] P. M. Melliar-Smith, L. E. Moser, and V. Agrawala. Broadcast Protocols for Distributed Systems. *IEEE Transactions on Parallel and Distributed Systems*, 1(1):17–25, January 1990.
- [70] D. Mills. Network Time Protocol (Version 3): Specification, Implementation, and Analysis. *DARPA Network Working Group*, RFC-1305, March 1992.
- [71] D. Mills. Improved Algorithms for Synchronizing Computer Network Clocks. *ACM SIGCOMM '94*, 24(4):317–327, September 1994.
- [72] J. Moy. Multicast extensions to OSPF. Request for Comments (Proposed Standard) RFC 1584, Internet Engineering Task Force, March 1994.
- [73] S. Navaratnum, S. Chanson, and G. Neufeld. Reliable Group Communication in Distributed Systems. In *Eighth International Conference on Distributed Computing Systems*, pages 439–446, San Jose, Calif., June 1988.

- [74] J. Nonnemacher and E. W. Biersack. Reliable Multicas: Where to use FEC. In *Proceedings of IFIP Fifth International Workshop on Protocols for High Speed Networks*, INRIA, Sophia Antipolis, France, October 1996.
- [75] J. Nonnemacher, E. W. Biersack, and D. Towsley. Parity-Based Loss Recovery for Reliable Multicast Transmission. In *ACM Sigcomm '97*, pages 289–300, Cannes, France, October 1997.
- [76] C. A. Noronha. Optimum routing of multicast streams. In *Proc. IEEE Infocom '94*, Toronto, Canada, June 1994.
- [77] C. Papadopoulos, G. Parulkar, and G. Varghese. An Error Control Scheme for Large-Scale Multicast. In *Proc. IEEE Infocom '98*, San Francisco, Ca., March 1998.
- [78] S. Paul, K. K. Sabnani, and D. M. Kristol. Multicast Transport Protocols for High Speed Networks. In *Proc. International Conference on Network Protocols*, pages 4–14, August 1994.
- [79] V. Paxson. Fast Approximation of Self-Similar Network Traffic. Technical Report LBL-36750, Lawrence Berkeley Laboratory and EECS Division, University of California, Berkeley, April 1995.
- [80] V. Paxson. An Introduction to Internet Measuring and Modelling. In *Tutorial 2, ACM Sigcomm '96*, Stanford University, August 1996.
- [81] V. Paxson and S. Floyd. Wide Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.
- [82] V. Paxson and S. Floyd. Why We Don't Know How To Simulate The Internet. Technical Report LBNL-41196, Network Research Group, Lawrence Berkeley National Laboratory, December 1997.

- [83] L. Peterson, N. C. Bucholz, and R. D. Schlichting. Preserving and Using Context Information in Interprocess Communication. *ACM Transactions on Computer Systems*, 7(3):217–246, August 1989.
- [84] S. Pingali, D. Towsley, and J. F. Kurose. A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols. In *Proc. 1994 ACM Sigmetrics and Performance '94*, pages 221–230, May 1994.
- [85] B. Rajagopalan. Reliability and Scaling Issues in Multicast Communication. In *Proc. Sigcomm '92*, pages 188–198, Baltimore, Maryland, August 1992.
- [86] S. Ramakrishnan and B. N. Jain. A Negative Acknowledgement with Periodic Polling Protocol for Multicast over LANs. In *Proc. IEEE Infocom '87*, pages 502–511, San Francisco, California, March 1987.
- [87] L. Rizzo. Effective Erasure Codes for Reliable Computer Communications Protocols. *Computer Communications Review*, 27(2):24–36, 1997.
- [88] H. Salama. Multicast Routing Algorithms: A Survey, January 1994. Unpublished Memorandum Available at <ftp://ftp.csc.ncsu.edu/pup/rtcomm/RTMulticast.ps>.
- [89] S. Schiff. Resolving Time-series Structure with a Controlled Wavelet Transform. *Optical Engineering*, 31(11):2492–2495, November 1992.
- [90] H. Schulzrinne. Voice Communication Across the Internet: A Network Voice Terminal. Technical Report UM-CS-1992-050, University of Massachusetts, July 1992.
- [91] H. Schulzrinne. Issues in Designing a Transport Protocol for Audio and Video Conferences and Other Multiparticipant Real-Time Applications. Technical Report draft-ietf-avt-issues-01, Internet Engineering Task Force (IETF), October 1993.

- [92] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. Technical Report draft-ietf-avt-rtp-06, Internet Engineering Task Force (IETF), November 1994.
- [93] N. Shacham. Multipoint Communication of Hierarchically Encoded Data. In *Proc. IEEE Infocom '92*, volume 3, pages 2107–2114, Florence, Italy, May 1992.
- [94] T. Speakman, D. Farinacci, S. Lin, and A. Tweedly. Pretty Good Multicast (PGM) Transport Protocol Specification. Technical report, Internet Engineering Task Force Internet Draft, January 1998. Available at <ftp://ds.internic.net/internet-drafts/draft-speakman-pgm-spec-00.txt>.
- [95] T. W. Strayer, B. J. Dempsey, and A. C. Weaver. *XTP: The Xpress Transfer Protocol*. Addison-Wesley Publishing Company, 1992.
- [96] R. Talpade and M. Ammar. Single Connection Emulation (SCE): An Architecture for Providing a Reliable Multicast Transport Service. Technical Report GIT-CC-94-47, Georgia Institute of Technology, Atlanta, Georgia, April 1995.
- [97] A. S. Thyagarajan and S. E. Deering. Hierarchical Distance-Vector Multicast Routing for the MBone. In *Proc. Sigcomm '95*, pages 60–66, Cambridge, Massachusetts, September 1995.
- [98] H. Tode, Y. Sakai, M. Yamamoto, H. Okada, and Y. Tezuka. Multicast routing algorithm for nodal load balancing. In *Proc. IEEE Infocom '92*, pages 2086–2095, Florence, 1992.
- [99] T. Turletti. H.261 Software Codec for Videoconferencing over the Internet. Technical Report 1834, Institut National de Recherche en Informatique et en Automatique (INRIA), January 1993.



- [100] L. Wei and D. Estrin. A comparison of multicast trees and algorithms. In *Proc. IEEE Infocom '94*, Toronto, Canada, June 1994. IEEE.
- [101] B. Whetten, T. Montgomery, and S. Kaplan. A High Performance Totally Ordered Multicast Protocol, August 1994. Research Memorandum Available at: <http://research.ivv.nasa.gov/projects/RMP/RMP.html>.
- [102] H. Zhang X. Rex Xu, A. Myers and R. Yavatkar. Resilient Multicast Support for Continuous-Media Applications. In *Seventh International Workshop on Network and Operating Systems Support for Digital Audiovisual (NOSSDAV '97)*, May 1997.
- [103] R. Yavatkar, James Friffoen, and Madhu Sudan. A Reliable Dissemination Protocol for Interactive Collaborative Applications. *ACM Multimedia 1995*, pages 333–343, November 1995.
- [104] R. Yavatkar and Leelanivas Manoj. Optimistic Strategies for Large-Scale Dissemination of Multimedia Information. *ACM Multimedia 1993*, pages 1–8, August 1993.
- [105] T. S. Yum and M. Chen. Multicast source routing in packet-switched networks. In *Proc. IEEE Infocom '91*, pages 1284–1288 (11B.2), April 1991.
- [106] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSeVation Protocol. *IEEE Network*, 7(5):8–18, September 1993.