Winter 2000

Multicast Services for Multimedia Collaborative Applications

Emad Eldin Mohamed Mohamed
Old Dominion University

MULTICAST SERVICES FOR MULTIMEDIA COLLABORATIVE APPLICATIONS

by

Emad Eldin Mohamed Mohamed
M.Sc. Computer Science, December 1996, University of Nebraska at Omaha
M.Sc. Computer Engineering, January 1994, Cairo University, Egypt
B.Sc. Electrical Engineering, August 1988, Cairo University, Egypt

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
December 2000

Approved by:

[illegible]
Hussein Abdel-Wahab (Director)

[illegible]
James Leathrum (Member)

[illegible]
Kurt Maly (Member)

[illegible]
Ravi Mukkamala (Member)

[illegible]
Christian Wild (Member)

ABSTRACT

MULTICAST SERVICES FOR MULTIMEDIA
COLLABORATIVE APPLICATIONS

Emad Eldin Mohamed Mohamed
Old Dominion University, 2000
Director: Dr. Hussein Abdel-Wahab

This work aims at providing multicast services for multimedia collaborative applications over large inter-networks such as the Internet. Multimedia collaborative applications are typically of small group size, slow group membership dynamics, and awareness of participants' identities and locations. Moreover, they usually consist of several components such as audio, video, shared whiteboard, and single user application sharing engines that collectively help make the collaboration session successful. Each of these components has its demands from the communication layer that may differ from one component to another. This dissertation identifies the overall characteristics of multimedia collaborative applications and their individual components. It also determines the service requirements of the various components from the communication layer. Based on the analysis done in the thesis, new techniques of multicast services that are more suitable for multimedia collaborative applications are introduced. In particular, the focus will be on multicast address management and connection control, routing, congestion and flow control, and error control. First, we investigate multicast address management and connection control and provide a new technique for address management based on address space partitioning. Second, we study the problem of multicast routing and introduce a new approach that fits the real time nature of multimedia applications. Third, we explore the problem of congestion and flow control and introduce a new mechanism that takes into consideration the heterogeneity within the network and within the processing capabilities of the end systems. Last, we exploit the problem of error control and present a solution that supports various levels of error control to the different components within the collaboration session. We present analytic as well as simulation studies to evaluate our work, which show that our techniques outperform previous ones.

ACKNOWLEDGMENTS

Of all the individuals lending their valuable guidance and assistance to this dissertation, Dr. Hussein Abdel-Wahab deserves the foremost acknowledgment. Without his great support and patience, this work could not be developed. Dr. Abdel-Wahab, as my advisor, successfully guided and advised me throughout the progress of this dissertation whenever I needed these guidance and advice.

My appreciation also goes to Dr. Leathrum, Dr. Maly, Dr. Mukkamala, and Dr. Wild for their contribution as members in my Ph.D. committee. I would like to thank them all for their valuable comments on my Ph.D. proposal, which I believe helped me to better shape my subsequent ideas for the dissertation. I would also like to thank them for their detailed review of the dissertation and for their comments.

TABLE OF CONTENTS

Page

LIST OF TABLES vii

LIST OF FIGURES viii

Chapter

I. INTRODUCTION 1
- 1.1 Background 2
- 1.2 Tasks of multicast communication 4
- 1.3 Objective 8
- 1.4 Approach 9
- 1.5 Outline 11

II. MULTIMEDIA COLLABORATIVE APPLICATIONS 13
- 2.1 Collaborative Systems Applications 13
- 2.2 Taxonomy of Collaborative Systems 15
- 2.3 Multimedia Collaborative Applications 16
- 2.4 Properties of Multimedia Collaborative Applications Components 17
- 2.5 Multicast Requirements of the Components of Multimedia Collaborative Applications 20
- 2.6 Summary and Conclusions 22

III. MULTICAST ADDRESS MANAGEMENT AND CONNECTION CONTROL 24
- 3.1 Introduction 24
- 3.2 Related Work 27
- 3.3 Addressing in the Internet 28
- 3.4 Performance Measures 29
- 3.5 Multicast Address Management for Local Sessions 30
- 3.6 Multicast Address Management for Global Sessions 34
- 3.7 Multicast Port Resolution 35
- 3.8 Performance Evaluation 41
- 3.9 Conclusion 46

IV. MULTICAST ROUTING 48
- 4.1 Introduction 48
- 4.2 Multicast Routing: the Challange 51
- 4.3 Taxonomy 53
- 4.4 Performance Indexes 54

4.5 Multicast Routing: A New Technique 55
4.6 Modeling the New Technique 61
4.7 Variations of the Routing Technique 64
4.8 Performance Analysis 66
4.9 Conclusion 75

V. MULTICAST CONGESTION AND FLOW CONTROL 76
5.1 Introduction 76
5.2 Principles and Techniques of Congestion Control 78
5.3 Multicast Congestion Control 80
5.4 Principles of Flow Control 83
5.5 Multicast Flow Control 84
5.6 Multicast Congestion and Flow Control: A New Technique 85
5.7 Performance Evaluation 89
5.8 Conclusion 94

VI. MULTICAST ERROR CONTROL 95
6.1 Introduction 96
6.2 Multicast Error Control 99
6.3 Performance Metrics 103
6.4 System Model and Simulation Measurements 104
6.5 Multiple Multicast Groups in ARQ 106
6.6 Multiple Multicast Groups in FEC 110
6.7 Multiple Multicast Groups in Hybrid Techniques 112
6.8 Multicast Error Control and Congestion and Flow Control 115
6.9 Conclusion 116

VII. CONCLUSIONS AND FUTURE WORK 118
7.1 Conclusion 118
7.2 Future work 121

REFERENCES 123

VITA 129

LIST OF TABLES

Table Page

1. Characteristics of multimedia collaborative applications components 20
2. Multicast requirements of multimedia collaborative applications 21
3. UDP port usage 36
4. Performance of address management techniques 45
5. Taxonomy of multicast port approaches 54
6. Messages used in the new routing technique 59
7. Performance of the routing techniques 75

LIST OF FIGURES

Figure Page

1. Application level multicast (S: source, D: destination, R router, L: link) 3
2. Network level multicast (S: source, D: destination, R: router, L: link). 3
3. Multicast services. 5
4. Multicast addresses for local sessions. 31
5. Address calculation based on Host ID. 33
6. Multicast addresses for global sessions in different network classes 34
7. Client server architecture of the port blocking experiments. 36
8. Port blocking probability as a result from the experimental work. 37
9. Port blocking probability as resulted from the analytical model. 40
10. Blocking probability when aggregating managers. 42
11. Blocking probability when increasing the size of the address space. 42
12. Minimum delay vs. minimum cost routing. 48
13. Source rooted routing. 49
14. Core rooted routing. 50
15. Taxonomy of multicast routing techniques. 53
16. Membership messages from destinations and sources to the RVP. 55
17. Multicast message from RVP to sources 56
18. Multicast tree for one source. 57
19. RVP flow chart. 60
20. Source flow chart. 61
21. Destination flow chart. 62
22. Router flow chart. 63
23. Average delay vs. network size. 67
24. Average delay vs. group size 67
25. Average bandwidth cost vs. network size. 68
26. Average bandwidth cost vs. group size. 68
27. Traffic concentration vs. network size. 69
28. Traffic concentration vs. group size. 70
29. Average bandwidth overhead vs. network size. 71

30. Average bandwidth overhead vs. group size. 71
31. Average memory usage vs. network size 72
32. Average memory usage vs. group size. 72
33. Average join latency vs. network size. 73
34. Average join latency vs. group size. 74
35. Congestion: incoming traffic ≥ outgoing traffic. 78
36. Congestion in unicast and multicast communications. 80
37. Multiple groups for layered data. 81
38. Hierarchical organization for destinations. 82
39. Data buffering and processing at destinations. 83
40. End system heterogeneity in unicast and multicast communications. 84
41. Grouping destinations according to their capabilities. 86
42. Data distribution over groups for real time transmission. 87
43. Data distribution over groups for reliable transmission. 87
44. System model. 90
45. Throughput in real time transmission 93
46. Example of FEC error control. 102
47. Hybrid techniques. 103
48. ARQ error control using multiple groups. 106
49. Latency in ARQ. 107
50. Destinations' overhead in ARQ. 108
51. Network overhead in ARQ. 109
52. Multiple Multicast groups in FEC. 110
53. Latency in FEC. 111
54. Destinations' overhead in FEC. 112
55. Network Overhead in FEC. 112
56. Multiple multicast groups in hybrid techniques. 113
57. Latency in the hybrid approach. 113
58. Destinations' overhead in the hybrid approach. 114
59. Network overhead in the hybrid approach. 114
60. Error control and congestion and flow control 115

CHAPTER I

INTRODUCTION

Multimedia collaborative applications are no longer unusual with the expectation of further growth of such applications. Driven by the trends towards teamwork and supported by the advances in computing and networking facilities, multimedia collaborative applications are gaining popularity as a solution that enhances collaboration among a geographically dispersed set of users. Multimedia collaborative applications exploit the computing and networking technologies along with the media acquisition and playback facilities to help a group of users, not necessarily residing in the same place, to work and interact together in a common task. This proves invaluable for many organizations such as large enterprises with sites located in different cities or even different countries. Examples of this class of applications include computer conferencing and distance learning [52].

Essential to the success of multimedia collaborative applications is efficient multicasting. Multicast is a communication mechanism that concerns data transfer among a set of end systems [23]. Many challenges face multicast communications—address management, connection control, routing, congestion control, flow control, and error control just to name a few. Indeed, a one-size-fits-all multicast is very doubtful. A multicast service that is suitable for distributed databases, for example, may not be the best for multimedia collaborative applications. This directs the research towards special case multicasts. For instance, many multicast error control techniques have been designed to handle specific types of applications [27], [65]. In order to provide a suitable multicast services for an application, it is more practical to design these services taking into consideration the application's specific properties and needs.

Multimedia collaborative applications have their particular characteristics. For example, a feature of these applications is that the number of participants in typical collaboration sessions is not very large (around 10 to 50 in most cases). Another feature

---

The journal model for this dissertation is the *IEEE/ACM Transactions on Networking*.

is that the changes in session membership (participants joining and leaving) are infrequent. Moreover, the identities and locations of the session participants are usually known. These features can be better demonstrated when contrasted with those of other multicast-based systems. For example, in news broadcast, the number of participants in a typical session is very large and the group dynamics is very fast. Also, most of the time, sources do not know the identities of the recipients nor do the recipients know each other.

Typically, a multimedia collaborative application consists of several components that, combined together, help make a collaborative session successful. For example, it may include video, audio, shared editor, shared whiteboard, and single user application sharing engines, all in one collaborative session. These components require an efficient multicast layer to disseminate information to each of the system participants. Each of these components, however, has its multicast needs that differ from those of the other components. For instance, a timely data delivery of audio and video streams is required, while a reliable data delivery is the main concern for shared editor or shared whiteboard.

The objective of this work is to provide a multicast layer that better serves multimedia collaborative applications. We first identify the characteristics of these applications and their individual components and we determine their multicast service requirements. Based on our analysis, we present new techniques for multicasting that take advantages of the special properties of multimedia collaborative applications to better serve their needs. We mainly investigate the problems of multicast address management and connection control, routing, congestion and flow control, and error control. We present new solutions for each of these problems. We present analytic as well as simulation studies to evaluate our work. Results of our studies show that our techniques outperform previous ones.

1.1 Background

As introduced earlier, multicast communications play a major role in the success of multimedia collaborative applications. Mainly, there are two techniques of multicast: application-level and network-level multicast. Multicast can be achieved at the application level through multiple unicasts from the source to the intended destinations. This assumes that the source knows the identities of the destinations. Application level multicast does not require any support from the network layer—all it requires is the support for unicast communications. This solution, however, suffers from three deficiencies: wasted communication bandwidth, wasted CPU computing resources at the sending end, and increasing delay between the last and the first receiving end systems, which is proportional to the number of receiving end systems.

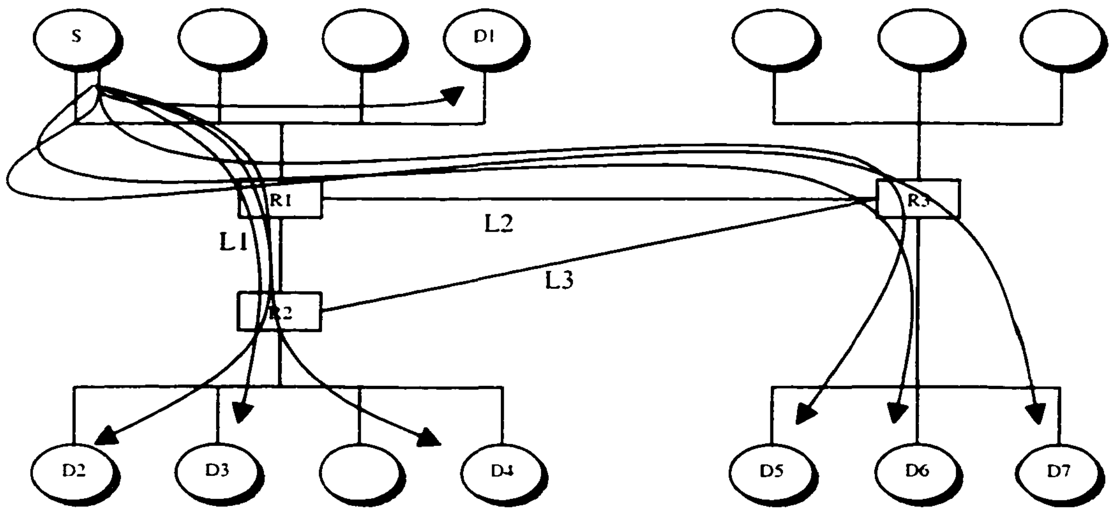

Fig. 1. Application level multicast (S: source, D: destination, R router, L: link).

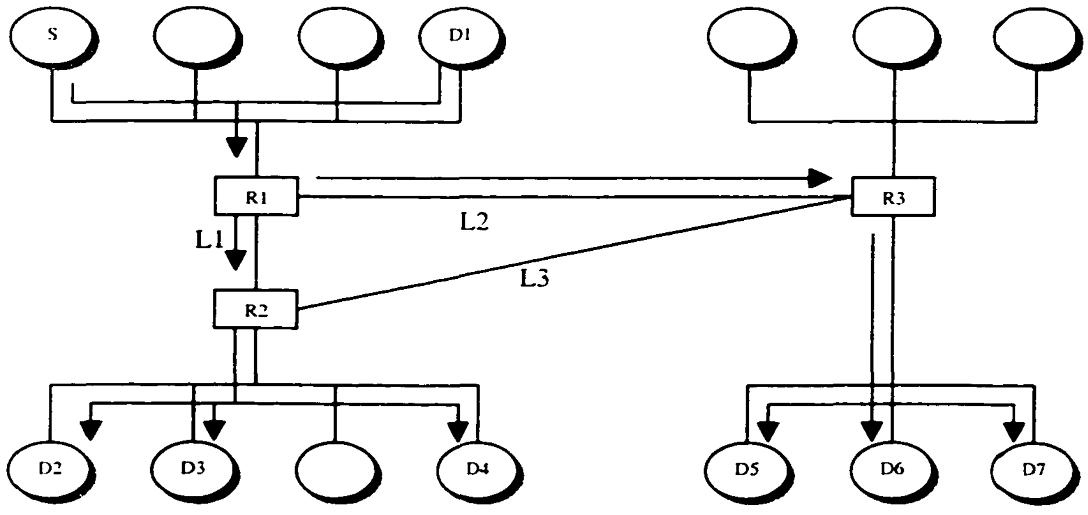

Fig. 2. Network level multicast (S: source, D: destination, R: router, L: link).

Fig. 1 demonstrates the use of multiple unicasts to achieve multicast. As Fig. 1 shows, the source **S** sends seven duplicates of the same packet for the intended destinations. Links **L2** and **L3** both carry three duplicates of the same packet. This overloads both the source—as it needs to process seven duplicates of the same packet—and the network links and routers—as they process and carry these duplicates. Moreover, if the source **S** sends the first copy to **D1** and the last one to **D7**, there is a delay between **D7** and **D1** in receiving the same packet due to the fact that **S** is processing five copies to the other destinations (**D2** to **D6**).

On the other hand, the network can support multicast. In this case, the source sends its multicast data to a multicast group. The network maintains information about the group members and forwards the multicast traffic to the intended destinations based on the information it has. This approach remedies most of the deficiencies in application level multicast. The source does not know about the destinations, it is all taken care of at the network layer. The source sends only one copy of the message, and the network delivers it only to the interested end systems. This way, messages are not duplicated over the communication links. There is no overhead at the source; the source does not keep information about the destinations and only one copy is sent. Also, no matter how many destinations there are, there is no such delay among the destinations due to the sequential order of sending copies of the same packet by the source—delay is only due to the network topology and the distribution of receivers in relation to the sender. Fig. 2 illustrates the case of network level multicast. For the rest of this dissertation, only network level multicast will be considered.

1.2 Tasks of multicast communication

The main difficulty that faces the network-level multicast approach is that the network layer must provide the suitable support to carry out the functionality required by multicast. Mainly, there are five tasks that should be incorporated in any multicast layer: address management, routing, congestion control, flow control, and error control. Fig. 3 gives these basic components and the relationship between them. These five tasks can be organized into two layers: network layer and transport layer. There are two components in the network layer: multicast routing and address management and there are three components in the transport layer: congestion control, flow control, and error control. As the figure shows, the tasks of address management and routing are independent, while there is a close relationship between the tasks of congestion control, flow control, and error control.

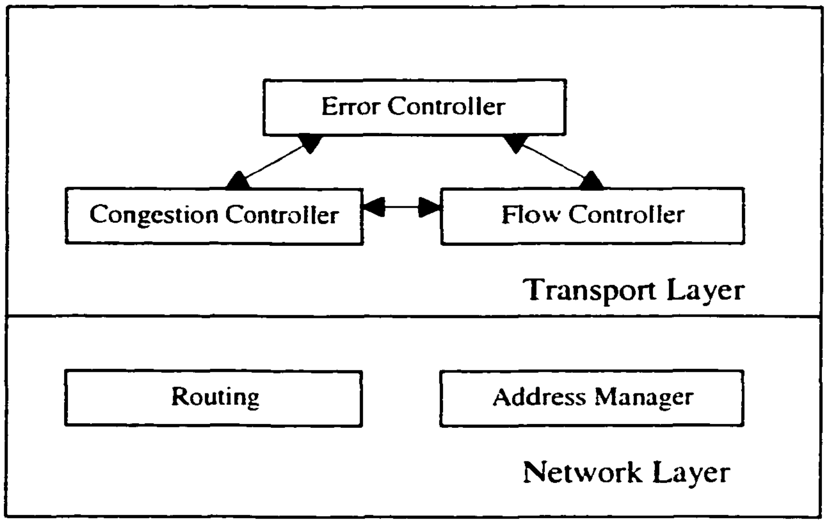

Fig. 3. Multicast services.

A multicast group is a collection of end systems that can be referenced as a single entity [12]. To allow simultaneous multicast sessions, multicast groups should be given unique identities and to allow multiple sessions within a host, service access points—named port numbers in many platforms—to the transport layer within the host are used. Thus, a multicast address is a combination of an address and a port number. The address is used to route the multicast traffic within the network to the host, while the port number is used to demultiplex communication traffic among multiple processes within the host. For an end system to receive a multicast traffic, it must join both the group address and port number. Thus, the problem of group identification can be broken into two sub-problems: multicast address management (the assignment of multicast addresses uniquely to multicast groups) and multicast port resolution (the agreement on a port number among all potential members of the group). A careless assignment of multicast addresses may result in address collision (two or more concurrent multicast sessions select the same address) or port blocking (an end system is blocked from joining the multicast session because the multicast port is already in use by another application within the end system). The basic challenge that faces multicast address management is the huge size of the environment—addresses should be assigned uniquely and dynamically to groups throughout the entire network. As for port resolution, the main difficulty is that potential group members are not assumed known. Also, the port usage within a host is dynamic; a port that is free by the start of the negotiation phase may not be so just after the negotiation has started.

Multicast routing is responsible for forwarding the multicast traffic from the source to the destinations. The problem, basically, reduces to building a tree—named a distribution tree—to route the data from the source to the destinations. Mainly, there are two types of trees: minimum cost tree and minimum delay tree. Finding the minimum cost tree is the Steiner tree problem and is proven to be NP-complete [36], [42], [43]. Building a minimum delay tree corresponds to finding the shortest path tree and is a tractable problem with many efficient algorithms [22]. Since many of today's applications require minimum delay trees, the goal is to efficiently build and maintain the shortest path distribution tree from a given source to a set of destinations. The main challenge that faces multicast routing is that group membership information is scattered within the network routers. To add to its complexity, many multicast group models do not require sources to join the group [17], [18]. For large inter-networks, the problem is how to relay the membership information only to the interested parts of the network in order to discover and maintain the shortest path distribution tree while group membership is dynamic and sources are not known.

We say that a network is congested when increasing the load does not increase the throughput of the network; rather, it dramatically decreases it. Congestion occurs when the traffic incoming at a node approaches or exceeds that of the outgoing. In such a case, the length of the queue at this node grows indefinitely. Since routers' queues are of finite lengths, some of the incoming traffic may get dropped. The problem gets even worse when the sources try to compensate for the lost messages and level up their transmission rates. In a best effort, packet switched network with no policy for traffic admission—the

Internet is an example—congestion control greatly affects the overall network performance. Since routers' reaction to congestion is limited, all end systems must cooperate to avoid congestion and to contain it would it happen. In unicast communication, a solution is to feedback the source with information about the received packets at the receiving end. Information about lost packets and packet delays can be used as indications of congestion along the path from the source to the destination. The source can then adapt to network congestion based on this feedback. However, the problem of congestion control is magnified in multicast communication since traffic originates from a single end system and is distributed along many links to many destinations. Some of these links may be congested while others may not, leaving the source with a problem to decide at which rate it should send. Moreover, sending a feedback from all destinations to the source may result in a feedback implosion at the source.

Flow control concerns regulating the data sent from the source so as not to overwhelm the destinations. The problem of flow control is similar to that of end-to-end congestion control since both try to regulate the data flow sent by the source. They differ, however, in their goals—congestion control tries to avoid overwhelming the network with data whereas flow control tries to smooth down the mismatch between the source and the destinations capabilities. Similar to congestion control, the problem of flow control in multicast is much tougher than that of unicast. Since there are many destinations with different capabilities, the source needs to decide which rate it should follow. Sending at the slowest destination rate may not be fair for faster ones. Also, forwarding feedback from all destinations to the source may cause a feedback implosion at the source.

In best effort networks, errors may occur. For example, data packets may get duplicated, received out of order, or even lost. Some applications, video for example, may perform reasonably well even with the existence of some errors. Some others, application sharing engines for instance, cannot tolerate any errors. The goal of the error control task is to efficiently correct the errors that may happen during data transmission. A common technique in unicast communications is to get the destination to feedback the source with the received packets. The source can use this information to detect errors and to correct these errors (using retransmissions). As in congestion control and flow control, the problem of error control in multicast is much harder than that of unicast. Since there are many destinations, adopting the approach of feedback from destinations to source is very expensive and a feedback implosion is the expected result. Also, different destinations may have different error rates and patterns. Directing retransmissions to the whole multicast group members may not be the best solution, since this may waste network bandwidth (to carry unnecessary retransmissions to those who did not experience errors) and overload destinations with unwanted retransmissions.

1.3 Objective

This work aims at providing multicast services for multimedia collaborative applications over very large inter-networks such as the Internet taking into consideration their special characteristics and needs. The problem at hand can be stated as the efficient delivery of multiple streams of data with varying requirements to a limited size group of well-known end systems with slow membership dynamics over a datagram packet-switched network.

While there are several techniques and protocols to provide multicast services, there remains a need for special-case multicast solutions that take advantage of the specific properties of multimedia collaborative applications and provide the efficient services required by the different components of these applications. Some of the challenges that face today's deployment of multicast include address management and connection control, routing, congestion and flow control, and error control. The existing techniques for handling these multicast issues are inefficient in most cases and insufficient in many others. It is the goal of this study to provide special-purpose solutions for the aforementioned multicast issues to provide multicast services based on the properties of multimedia collaborative applications and to meet the specific demands of the different components of these applications.

1.4 Approach

To achieve the goal stated above, this work first identifies the overall characteristics of multimedia collaborative applications that make these applications different from other multicast based applications. Then, the most common components in typical multimedia collaborative applications are explored and the multicast needs of each of these components are identified. The impact of the characteristics of these applications over multicast is investigated. Mainly, the impact of group size, group dynamics, and awareness of group participants over multicast is studied. Based on this study, new designs for some of the multicast services that better match the special needs of the different components are proposed. In particular, each of the multicast five tasks introduced earlier is explored from the multimedia collaborative applications point of view.

First, we investigate multicast address management and connection control. We notice that multicast sessions can be classified according to their geographical span into two types: local sessions and global session. In many multicast sessions, the participants of the sessions are located within a single domain. For this type of multicasting, requiring a unique address over the global inter-network is just wasting of resources and efforts, since it is sufficient to make sure that the address is unique within the domain where the members exist. Adopting such assumption simplifies the address management task and greatly increases the number of multicast addresses available for local sessions, since the same address can be safely used in different domains at the same time. On the other hand, many multicast sessions do not restrict their members to exist within a single network; rather, members are scattered over the entire inter-network. For this type of multicast, selecting a unique global multicast address is required to prevent cross talk between the different sessions. Techniques that may work reasonably well in local sessions may not perform the same in global ones.

We provide alternative techniques for multicast address management (local and global sessions) and we analytically study the performance of each technique and compare it against the performance of previous ones, demonstrating the advantages and disadvantages of each. As for port resolution, we suggest reserving a number of ports for multicast use only. This suggestion is justified recognizing the rapid use of multicast applications. Having a number of ports reserved for multicast use removes the competition of UDP in using these ports, which is the main reason for port blocking. To minimize the competition from other multicast sessions, we calculate the port number as a function of the multicast address (A simple hashing function can be used for this purpose). Any process that wishes to join the multicast group can calculate the port number knowing the multicast address. This way, we remove the need for a central server to manage port assignment and its complication. This technique still suffers from a port blocking in case there is collision in the hashing function. The probability of such collision, however, can be minimized by reserving a larger number of ports for multicast use.

Second, we study the problem of multicast routing. Multicast routing techniques can be classified into two types: broadcast and prune and shared tree. Protocols based on the broadcast and prune technique provide the shortest path tree from the source to the destinations. The main disadvantage, however, is the waste of the network bandwidth in building and maintaining the tree due to the periodic broadcast of multicast data. On the other hand, shared tree routing is efficient in building and maintaining the tree. It, however, may not give the shortest path tree for some sources and may result in traffic concentration around the tree core.

We introduce a new approach that fits the real time nature of multimedia applications by building the shortest path trees from the sources to the destinations. The new techniuqe combines the advantages of both broadcast and prune and shared trees approaches while avoiding their shortcomings—it provides the shortest path tree, yet it avoids the large bandwidth consumed in building and maintaing the tree. Basic assumptions for our technique to work is small size and slow membership dynamics of the multicast groups, which are typical in multimedia collaborative applications. We present a simulation study to compare the performance of our technique against broadcast and prune and shared tree techniques. Results of the simulation show that our approach outperforms both techniques.

Third, we explore the problem of congestion and flow control and we introduce a new algorithm that takes into consideration the heterogeneity within the network and within the processing capabilities of the end systems. In this algorithm, destinations are organized into multicast groups according to the bandwidths of the links from the source to the individual destinations and according to the computing power of the destinations. The source maintains a window for each group, which regulates the transmission rate for the group. In order to avoid feedback implosion, a representative is assigned for every group and is responsible to send its feedback to the source to advance the group's window. We investigate the problems of group splitting, merging, and migration and we provide solutions for them. We introduce an analytic study to evaluate our algorithm. Results of our study demonstrate the superiority of our algorithm over other techniques.

Last, we exploit the problem of multicast end-to-end error control. Mainly, there are two approaches for error control: automatic repeat request (ARQ) and forward error correction (FEC). ARQ relies on error detection and retransmission while FEC sends correction codes that can be used by the receiving end to recover from. While ARQ suffers from large delays, FEC faces the problem of determining the suitable amount of correction codes. Recently, hybrid approaches have emerged to combine the advantages and avoid the disadvantages of both techniques. We notice that different components of multimedia collaborative applications require different levels of error control. We present a solution that is based on using multiple multicast groups and supports various levels of error control to the different components within the collaboration session. Specifically, we enhance an early work in utilizing multiple groups in ARQ and we introduce new techniques for using it in FEC and hybrid approaches. In our work, we consider the effect of multicast routing over error control; an important factor that has been ignored by previous studies. We present a simulation study to evaluate our work, which gives encouraging results compared against other techniques.

1.5 Outline

The rest of this dissertation is organized as follows. Chapter II is an overview of the multimedia collaborative applications. It first presents taxonomy of collaborative applications and introduces some of their popular systems. Then it investigates multimedia collaborative applications and explores their general characteristics. Last, it presents the multicast requirements for these applications. Chapter III investigates the multicast address management and connection control problem and introduces a new technique for it. An analytical performance evaluation study is introduced and is used to compare the different address management techniques. Chapter IV discusses multicast routing. In this chapter, a new approach for multicast routing is introduced. A simulation study for performance evaluation is presented and is used to compare the different routing techniques. Since the problems of end-to-end congestion control and flow control are related, Chapter V investigates both problems. The chapter presents a new algorithm based on using multiple multicast groups to control congestion and to regulate flow over a large heterogeneous inter-network with destinations of varying capabilities. An analytical study is presented and is used to evaluate the performance of the new technique. Chapter VI explores the error control issue in multicast. A scheme that is based on using multiple multicast groups is used to control error in multicast communications. A simulation study is presented to evaluate the performance of the new technique. The conclusion and the future work of this work are given in Chapter VII.

CHAPTER II

MULTIMEDIA COLLABORATIVE APPLICATIONS

Driven by the trends towards teamwork and supported by the rapid advances in computing and networking facilities, collaborative systems are taking their place side by side with single user systems. A collaborative system can be defined as a system that allows a group of users, usually geographically dispersed, to work together in harmony on a common task using computing and networking facilities [25]. Collaborative systems are different from traditional distributed systems such as distributed databases in the way each system handles the actions of its users. While distributed systems strive to give each of their users the illusion that she is working alone, a main feature of collaborative systems is to make each of their users aware of the existence of the others and of their actions [31].

Many multi-user applications require multicast services to disseminate information among their participants. The multicast demands of different applications may vary from one to another. Multimedia collaborative applications are examples of systems that have special characteristics and different multicast requirements. This chapter starts with an overview of selected collaborative systems. Taxonomy of collaborative applications is given. Then multimedia collaborative applications class is introduced as a special class of collaborative systems. The overall characteristics and the properties of the individual components of multimedia collaborative applications are investigated and the multicast requirements for such applications are presented.

2.1 Collaborative Systems Applications

For the last two decades many collaborative systems have been introduced. Examples of these systems include computer conferencing, chat applications, electronic meeting systems, co-authoring systems, and data sharing systems [32]. The following is a brief discussion of these systems.

Computer conferencing is a class of systems that allows its users to post information to be accessible to others, very similar to a bulletin board. The information sent can be text, video, or audio. A well-known example of a text-based computer conferencing is the *Usenet*, which runs over the Internet. Usenet allows several addressable newsgroups to exist. Users can post to and read from the newsgroups that meet their interest. A video-based computer conferencing, also known as video conferencing, is a system that allows users to view their moving pictures at the same time. A well-known example of a video conferencing system is *vic* (video conferencing) which runs over the Internet and allows multiple users to transmit their videos and see others' in small windows on their computer screens [53]. Similar systems have been developed for audio. For instance, *vat* (visual audio tool) is an audio conferencing system that also runs over the Internet and allows its users to send their audio and receive the others' [38].

Chat applications are text-based systems that allow simultaneous discussions among set of users. An early chat system is the Unix *talk* system, which allows two users to interact. A more recent system is the Internet Relay Chat (*IRC*). IRC allows multiple users to interact simultaneously in the sense that whatever a user types, the other users can read instantaneously.

Electronic meeting systems have been developed to enhance the way people hold their meetings. There are many uses for electronic meeting systems. One example is the electronic voting system found in many parliaments, which provides a rapid and anonymous way of casting votes. Another example is brainstorming, where several colleagues meet to develop and form ideas. An example of an electronic meeting system is *colab* [76], which is developed by *Xerox PARC*. The system contains a large screen connected to a set of personal computers or terminals that are located in a single room. Users can use the computers for typing and the large screen displays what the users type.

Co-authoring systems are text-based systems that allow a group of authors to edit the same document at the same time. Authors may edit different parts of the documents, or they may watch what the others are typing. This leads to the fact that authors may need to scroll to different parts of the documents without affecting the others. This in turn may require the existence of several scroll bars equal to the number of the current active authors so that each author can be aware of what parts of the documents the others are working in. An example of co-authoring systems is *Quilt* [51].

Data sharing systems can be further divided into two categories: collaboration aware and collaboration transparent. Collaboration aware systems are built taking collaboration into account. Examples of these systems include shared whiteboards, shared editors, and shared drawing systems. Collaboration transparent systems, also know as collaboration naïve, share already existing single user application systems. Several sharing engines have been developed to share *X* Window, Microsoft Windows, and Java applications. Examples include X Teleconferencing and Viewing (*XTV*) [1], *NetMeeting* [54], and Java Collaborative Environment (*JCE*) [2].

2.2 Taxonomy of Collaborative Systems

Collaborative systems can be categorized based on many criteria. A taxonomy of collaborative systems can be better understood when demonstrated by a multidimensional space, with each criterion as a one dimension in that space. A particular state of a collaborative system class can be thought of as one point in that space. It is worth noting that a collaborative system may fall in different categories in different situations depending on the way the system is used. Criteria upon which a collaborative system can be categorized include temporal distribution, spatial distribution, interaction, coordination, and visualization. The following is an overview of these criteria [68].

A collaborative system may allow its users to collaborate at the same time or at different times. The first category is called synchronous while the latter is called asynchronous. Examples of synchronous systems include shared whiteboards, application sharing engines, and video and audio conferencing. A text-based computer conferencing system is an example of an asynchronous system.

Depending on the physical location of the participants of a collaborative system, the system can be classified as distributed or not. A distributed system has its participants work from different places.

Interaction can be implicit or explicit. For example, using shared text or shared drawing can be considered as an implicit interaction, while using video or audio is an explicit interaction.

Having multiple users working on shared objects concurrently requires coordination among them, otherwise inconsistency and confusion may result. Some systems, such as those involving small groups, may require less coordination than other systems. For example, in brainstorming, every user should have the same access to the shared objects with less coordinator intervention. Other systems may require tight coordination, which involves a turn taking mechanism. Example of these systems include audio conferencing and some single user application sharing engines such as XTV.

One paradigm of visualization in collaborative systems is What You See Is What I See (*WYSIWIS*). In a strict WYSIWIS, all users see the same thing at the same time. There are, however, relaxed WYSIWIS paradigms where users may have the flexibility to get different views of the same object. For example, authors in a shared document may like to edit different parts of the same document at the same time.

2.3 Multimedia Collaborative Applications

An emerging class of collaborative systems is multimedia collaborative applications. Examples of this class are teleconferencing and distance learning [52]. Generally, multimedia collaborative applications are characterized by the following:

1. Involve several media
2. Synchronous
3. Spatially distributed
4. Possess implicit and explicit interaction
5. Coordination needs vary from one component to another
6. Visualization varies from one component to another

Multimedia collaborative applications consist of several components that collectively help make the collaborative session successful. Beside text, graphics, and still images, other media such as audio and video are incorporated to add more interactions among the participants within the collaborative system. The advances in media acquisition and playback devices help make the integration of these media within collaborative systems successful. Many audio and video applications are interactive, in the sense that the delay encountered from the time one user submitting audio or video data to the time other users receiving them is acceptable considering the human perception of audio and video media.

Multimedia collaborative applications are synchronous, in the sense that the participants of a collaborative session work at the same time. This can be contrasted with asynchronous multimedia collaborative systems where the system users need not work at the same time. For example, the properties of a distance learning system [52], where the instructor communicates with his students live are different from those of a telelearning system [4], where the instructor records his class to be accessed later by the students. Because of time delay constraints, intra and inter media synchronization in synchronous systems is more challenging than in asynchronous ones [77].

Normally, participants of the multimedia collaborative applications are located in different places. This adds to the requirements of more interaction and coordination mechanisms among the participants. Audio and video can support explicit interaction among participants. Also implicit interaction can be provided through the use of other shared tools like a shared whiteboard. Coordination, on the other hand, varies from one component in the multimedia collaborative applications to another. A floor control mechanism may be necessary in some components such as single user application sharing engines [3], whereas a free-floor may be sufficient in other components like a shared whiteboard.

The visualization requirements vary from one component in the multimedia collaborative applications to another. While some components, such as a shared whiteboard, may require strict WYSIWIS, a relaxed WYSIWIS may be required in others, such as a shared editor and shared drawing.

2.4 Properties of Multimedia Collaborative Applications Components

Multimedia collaborative applications possess some characteristics that render them different from other multicast-based systems. Also, these applications usually consist of different components that by themselves have properties that may differ from one component to another. Typical components of these applications are video, audio and data sharing tools such as shared whiteboard, and single user application sharing engines. Moreover, a multimedia collaborative application usually incorporates a group manager and floor controller to take care of the group and floor management issues. This section investigates, from the multicast communication view point, the overall properties of multimedia collaborative applications as well as the characteristics of the individual components comprising these applications. For the rest of this section, we base our study on XTV [1] and JCE [2] as two examples of application sharing engines.

2.4.1 Overall Characteristics of Multimedia Collaborative Applications

Mainly, there are six issues that are of special importance when dealing with the multicast communication aspects of multimedia collaborative applications:

1. Group type
2. Group size
3. Group dynamics
4. Number of senders
5. Participants' identities and locations
6. Interaction

Usually, the multicast groups of multimedia collaborative applications are closed, meaning that the senders to the groups are themselves members of these groups. For example, in a desktop teleconferencing system all participants are known and only those participants can send to the group. This can be contrasted to open groups that exist in other systems where senders need not be members to send to the group.

A very important feature of multimedia collaborative applications is the group size. The typical number of participants in a collaborative session is not very large. As an example, the number of participants in a desktop conferencing system is around 4 or 5 and the number of participants in a distance learning system is similar to a typical classroom size, which is around 30. These numbers are very small when compared to other systems such as news broadcast where the number of participants may well exceed tens of thousands.

Normally, the group dynamics of multimedia collaborative applications is not fast, that is the frequency with which participants join and leave the collaborative session is not high. Normally, all participants join by the start of the session and leave at its end. This can be contrasted to systems of fast dynamics such as news broadcast, where participants join and leave the group very frequently.

In multimedia collaborative applications, it is very common that all the session participants are by themselves potential senders to the group. This is in contrast to systems where there is only one sender in the group.

Participants of multimedia collaborative applications usually know the identities of each other. Also, the participants of a typical collaboration session may be located in different places, that is, a typical session may span a large inter-network such as the Internet.

Most of the multimedia collaborative applications components are interactive, in the sense that the receiving ends play an active role that may change the subsequent actions of the senders during the course of the collaborative session. Interactivity can be better demonstrated when contrasted with a passive destination that has no effect on the way the sender sends its data. The interactivity characteristic imposes some delay constraints on the data transmission, which may be affected by the human perception of the different media. One way to minimize this delay is to interleave the playback with the receiving of data rather than buffering the whole data then playing it back.

2.4.2 Characteristics of Multimedia Collaborative Applications Components

As introduced before, multimedia collaborative applications are usually composed of several components, each has its own properties. In the following, the communication characteristics of the commonly used components are investigated based on two issues: volume of data and sensitivity to errors and packet loss. TABLE 1 summarizes the characteristics of the most common components in multimedia collaborative applications.

TABLE 1

CHARACTERISTICS OF MULTIMEDIA COLLABORATIVE APPLICATIONS COMPONENTS

Component	Volume	Sensitivity
Audio	Very large	Tolerable
Video	Huge	Very tolerable
Whiteboard	Very small	Sensitive
JCE	Small	Very sensitive
XTV	Large	Extremely sensitive
Group manager	Small	Sensitive
Floor controller	Small	Sensitive

The volume of data to be transported varies from one component to another. On the one hand, some applications, namely video, audio, and XTV, involve large amounts of data. The amount of data involved in video is much larger than that in audio, which in turn is larger than what is found in XTV. On the other hand, applications like a shared whiteboard and JCE deal with smaller amounts of data.

Errors in the form of data loss or duplication may happen at the receiving end. The response of components to errors varies from one component to another. Video, audio, and shared whiteboard tolerate errors at different levels. Based on the human perception to video and audio, it is found that video is less sensitive to errors than audio. Shared whiteboard can survive some errors, but is more sensitive to errors than audio and video. On the other hand, some components like JCE and XTV are very sensitive to errors.

2.5 Multicast Requirements of the Components of Multimedia Collaborative Applications

Having introduced the overall characteristics of multimedia collaborative applications and the properties of their individual components, this section investigates the demands of these applications from the communication layer. Knowing the properties of the applications and their multicast demands can help in designing multicast services that take advantages of these properties and provide the suitable functionality required. There are four issues concerning the multicast demands of multimedia collaborative applications: volume of data, nature of this data, flexibility in bandwidth requirement, delay constraints, and reliability. TABLE 2 summarizes the multicast requirements for the most common components in multimedia collaborative applications.

TABLE 2

MULTICAST REQUIREMENTS OF MULTIMEDIA COLLABORATIVE APPLICATIONS

Component	Volume	Transmission	Bandwidth	Delay	Reliability
Audio	Very large	Continuous	Flexible	Constrained	Best effort
Video	Huge	Continuous	Flexible	Constrained	Best effort
Whiteboard	Very small	Bursty	Not flexible	Relaxed	Semi-reliable
JCE	Small	Bursty	Not flexible	Relaxed	Reliable
XTV	Large	Bursty	Not flexible	Relaxed	Reliable
Group manager	Small	Bursty	Not flexible	Relaxed	Reliable
Floor controller	Small	Bursty	Not flexible	Relaxed	Reliable

From the previous section, some applications involve larger amounts of data than others. A consequence of this is those applications that involve larger data volume require more communication bandwidth than the others. Video and audio, for example, require more bandwidth than other applications like JCE and shared whiteboard.

Data transported may be bursty or continuous in nature. For example, video and audio may require continuous data transmission while shared whiteboard, JCE, and XTV are bursty in nature.

The flexibility of communication bandwidth requirement varies from one component to another. While some components require that all their data be sent and received to operate, others may work even if some of their data is discarded. Audio and video, on the one hand, may operate reasonably, at a less quality however, even if the allocated communication bandwidth is less than the bandwidth required for transmitting the whole data. XTV and JCE, on the other hand, must have all their data transmitted to function properly.

Delay constraint varies from one application to another. While video and audio require the delay to be very small, shared whiteboard, JCE, and XTV may have relaxed delay constraints.

The requirements for reliable data delivery for the different components are based on the applications' sensitivity to errors and packet loss discussed in the previous section. While XTV, JCE, and shared whiteboard require reliable transmission, audio and video may allow best effort data delivery in order to meet the low delay requirements.

In fact, for the same application, these requirements may vary from one operation to another. For example, a shared whiteboard requires reliable transmission. A small loss of data may degrade the quality of the application, yet the application can be used despite this quality degradation. The same argument holds for XTV and JCE. XTV, for example, is the one of the strictest systems when it comes to reliability. Yet, different operations within XTV itself can be treated differently. Some operations require a full reliable transmission, while others may not have the same reliability constraints. For example, creating a window is an operation that requires full reliability, whereas displaying an image within that window may succeed even in the presence of data loss (the image displayed will be of less quality, yet the application will not crash) [77]. There is no doubt that the quality of the session would be affected when allowing best effort transmission, yet the overall performance could be enhanced.

2.6 Summary and Conclusions

In this chapter we have presented an introduction to collaborative systems. We then have presented multimedia collaborative applications as a special class of collaborative applications that include multiple media such as audio, video, and textual data. We have investigated the overall characteristics of multimedia collaborative applications. A typical multimedia collaborative application has few participants and usually the participants span a large inter-network. The dynamics of the collaboration session is slow. Moreover, a multimedia collaborative application consists of several components that help make the collaboration session successful. Examples of these components include audio, video, single user application sharing engines, and shared whiteboard. We have presented the characteristics of each of these components and we have introduced their demands from the communication layer. Based on the analysis presented in this chapter, new techniques for multicast services will be introduced in the next chapters.

CHAPTER III

MULTICAST ADDRESS MANAGEMENT AND CONNECTION CONTROL

As introduced earlier, multicast is a communication mechanism that concerns data delivery to a group of processes. To allow simultaneous multicast sessions, multicast groups should be distinguished from each other in order to deliver multicast traffics only to their proper destinations. One of today's largest and most widely used networks is the Internet [67]. The Internet identifies multicast groups by a combination of multicast address and port number. It, however, does not control multicast address assignment, nor does it handle port resolution. Many techniques have been introduced for these two problems. In this chapter, we examine multicast address management and port resolution in large inter-networks such as the Internet. First, we notice that multicast sessions can be classified according to their geographical span into two types: local sessions and global sessions. We provide alternative techniques for multicast address management (local and global sessions). Moreover, we study the problem of multicast port resolution and we present a mechanism for it. We study the performance of each technique and compare it with the performance of previous ones, demonstrating the advantages and disadvantages of each.

3.1 Introduction

Multicast is a communication mechanism that concerns data delivery to a set of processes [12]. The multicast model used in the Internet is the host group [18]. In this model, processes interested in receiving a multicast traffic are organized into a multicast group. To allow simultaneous multicast sessions, multicast groups are given unique identities. Destinations must join the group to receive its multicast traffic. Groups may have any number of members that may be located anywhere in the Internet. Group membership is dynamic—processes can join the group and members can leave at any time—and there is no negotiation performed between processes in order to join or leave the group. Sources, on the other hand, need not join the group, nor need they know the identities of the group members. only they need to address the group by its identity [14]. [23]. [56].

The current implementation of the Internet identifies multicast groups by a combination of a four-byte address and a two-byte port number [14], [69]. Multicast addresses are used to route the multicast traffic to its intended hosts within the network, while port numbers are used to demultiplex communication traffic among multiple processes within a host. The current implementation of the Internet does not handle multicast address assignment. nor does it control the selection of port numbers. Without an authority to manage these two tasks, two anomalies may result: address collision and port blocking [24], [64].

Address collision arises when two groups are assigned the same multicast address. In this situation, the network routers have no means to distinguish between the two groups and multicast traffic intended for either group will be delivered to both. This problem. also known as cross talk, is not desirable since it wastes network bandwidth and processing efforts in the groups' members in delivering and processing the unwanted traffic of the other group. The increase in traffic is considerable recognizing that many of the multicast applications incorporate audio and video, which involve huge amount of data.

Moreover, if two groups select the same address and port number, the multicast traffic destined to either group will be delivered to the application layers of both. This may result in undefined behavior of the applications as they receive out of context data. To overcome this problem, filters are needed in the applications to get rid of any unwanted traffic. This is yet another overhead that could be avoided should the multicast address/port combinations be selected uniquely.

The second anomaly is related to the selection of port numbers. Within a host, port numbers can be considered as a limited resource that is shared among many processes. In general, if a port is allocated to a process, it cannot be allocated to another one at the same time. That is, for a process to join a multicast group the port number of the group must be free in the process' host, otherwise the process will not be able to join the group until the port is freed. This problem is referred to as multicast port blocking.

The current implementation of the Internet relies on the assumption that the probability of address collision and port blocking is negligible and does not pay attention to either problem. Moreover, IPv6 (version 6 of the Internet protocol) increases the multicast address significantly, which in turns reduces the probability of collision [21]. The increasing use of the Internet—increasing the address space in IPv6 also has the potential of increasing the number of hosts in the Internet—and the development of many multicast applications—examples include teleconferencing and distributed simulations—suggest that this assumption may be invalid in the near future.

The problem at hand can be divided into two related parts: address management and port resolution. Multicast address management is the assignment of multicast addresses uniquely to multicast groups, whereas port resolution is the agreement on a port number among all potential members of the group. Multicast address management can be viewed as a problem of managing distributed shared resources—the multicast addresses. The basic challenge that faces multicast address management is the huge size of the environment—addresses should be assigned uniquely and dynamically to groups throughout the entire Internet. As for port resolution, the main difficulty is that potential group members are not assumed known to each other. Also, port usages within hosts are dynamic: a port that is free by the start of the negotiation phase may not be so just after the negotiation has started.

The problem of multicast address management has attracted the attention of many researchers and several techniques have been introduced for both address management and port resolution. In this chapter, we investigate multicast address management and port resolution in large inter-networks, and we take the Internet as an example. We notice that multicast sessions can be categorized into two classes according to their geographical span: local sessions and global sessions. Group members in local sessions are restricted to exist within a single domain, whereas in global sessions they may span the entire inter-network. For local sessions, a unique address within the domain is sufficient (that is, there is no need to have a unique address throughout the global network). For this case, we propose techniques that avoid or detect and recover from multicast address collision. For global sessions, we present a technique that calculates multicast addresses as a function of the network ID to avoid address collision. As for port resolution, first, we suggest reserving part of the port space to multicast communication to reduce the probability of port blocking. Second, we introduce a technique to calculate port numbers as a function of the group address to allow processes to get the port number of a group knowing its address. We evaluate our techniques and compare them with the previous ones. Results of our study show superiority of our techniques.

The rest of this chapter is organized as follows. Section 3.2 discusses some of the related work in this field. Addressing in the Internet is the topic for Section 3.3. Section 3.4 presents the performance measures used to evaluate different multicast address management and port resolution techniques. Address management for multicast local sessions is introduced in Section 3.5. Section 3.6 is devoted for address management for global multicast sessions. Section 3.7 discusses the multicast port resolution problem. Evaluation of our techniques is given in Section 3.8.

3.2 Related Work

Recently, three techniques have been introduced to approach multicast address management and port resolution: multicast group authority, address space partitioning, and extended address. The first technique assigns multicast addresses by an outside authority: the Multicast Group Authority (MGA), which is tree structured [14]. Every node in the tree is assigned a block of multicast addresses. The size of the block depends on the location of the node within the tree, with the root of the tree controls the entire multicast address space. When a node receives a request for a multicast address, it assigns it an address if it has one and marks it unavailable until the end of the multicast session; otherwise it propagates the request upward along the tree. If the root of the tree runs out of addresses, a request is propagated downward the tree to reclaim the unused addresses. This technique eliminates the problem of address collision. It, however, suffers from two drawbacks: large setup delay and address blocking—a denial for multicast address requests as a result of exhausting the multicast address space. Moreover, it does not discuss the port blocking problem.

The second technique is to partition the multicast address space and assign every network a fixed number of addresses [24]. Every network is required to run one process to manage the assignment of multicast addresses within the network. This technique removes the problem of address collision and it eliminates the large setup delay encountered in the MGA technique discussed above. However, it suffers from a higher address blocking probability; a network may exhaust its address space while others may have many addresses available. Also, as in the MGA, this technique does not handle port blocking.

Another technique proposes the use of virtual ports to manage both address and port assignment. In their paper [64], Eleftheriadis et al. have noticed that port number is an integral part of the group identity and have proposed an extended multicast address algorithm for calculating the group address (6 bytes) out of the unicast IP address (4 bytes) of the host initiating the multicast session. Each process calculates its multicast address and the scheme ensures the uniqueness of the extended calculated multicast address. The algorithm greatly lowers the address blocking probability by increasing the multicast address space. Since routing is based solely on the 4-byte multicast address, the extended address may not work at the network level and cross talk may still result if two groups select the same first 4 bytes.

To resolve the port blocking problem, Eleftheriadis et al. have introduced the notion of virtual ports, which distinguishes between the multicast port (the virtual port) and the port the process uses in receiving multicast traffic (the actual port) [64]. When a process joins a multicast group, it needs not join the virtual port, instead it joins any free port in its host. Each host is required to maintain a table to map between the virtual and the actual ports, and when receiving multicast traffic destined for the virtual port, it delivers it to the actual port joined by the process. This scheme, however, needs a system support and suffers from the overhead encountered in maintaining the map table.

3.3 Addressing in the Internet

One of today's most famous network protocols is the Internet protocol (IP) [67]. In IP version 4 (IPv4), each address is a 32 bit long. Historically, the Internet had 5 address classes: A, B, C, D, and E. Classes A, B, and C were for unicast addressing, class D was for multicast, and class E was reserved for future use. This addressing mechanism proved inefficient with regard to address space usage. Recently, a classless scheme has been introduced to meet the increasing demands for unicast addresses [28]. In this scheme, there are no boundaries between classes A, B, and C. An IP address consists of two entities; network ID and host ID (a site can opt to divide its host ID into two parts: subnet ID and host ID). In classless addressing, every network is assigned a 32-bit address and a 32-bit mask. Bits set to one in the mask cover the network ID within the address, whereas those of zero value cover the host ID (bits of 1 in the mask start from the leftmost and are always contiguous). This scheme makes use of many of the wasted addresses using the original class-based technique. As for multicast addressing, IPv4 still uses class D, which has "1110" as the high order bits ranging from 224.0.0.0 to 239.255.255.255, to multicast communication. This range spans only 1/16 of the entire address space.

Similar mechanism has been adopted for IP version 6 (IPv6) [21]. The length of the address in IPv6 is 128 bits. There are no address classes in IPv6; unicast addresses are aggregatable with contiguous bit-wise masks very similar to classless IPv4 addresses. As for multicast, the first leftmost 8 bits in the address are always set to 1, the next 8 bits are reserved for flags and scope, and the remaining 112 bits are for the group ID. This range spans 1/256 of the entire address space.

3.4 Performance Measures

The ultimate goal of a multicast address management technique is to eliminate address collision and port blocking and to perform well according to certain measures. Of major importance of these measures of performance are *probability of address blocking*, *address acquisition delay*, and *processing and communication overhead*. Since many of the approaches introduced for port resolution may result in port blocking, the *probability of port blocking* is also of special interest.

Address blocking is the denial of multicast address requests as a result of exhausting the managed multicast address space. In many situations, and due to a bad management policy, address blocking may result despite the availability of multicast addresses. For example, address space partitioning may cause address blocking in one network as a result of exhausting the block assigned within it, while some other networks may have plenty of addresses available.

The second measure is address acquisition latency: the time elapsed between issuing a request for a multicast address until getting the address. For example, the MGA may cause a high latency especially when the root of the tree runs out of addresses. In this case, a multicast address request is delayed until addresses are reclaimed and redistributed to the tree nodes. As for port resolution, the latency is the time consumed by the resolution process until a decision has been made on which port should be used for the multicast session.

The third measure is the processing and communication overhead: the overhead experienced by the management entities and the communication facilities in conducting the management process. This overhead is considerable when having a central server managing the multicast addresses. Of particular importance is the load placed on the server to handle all address requests. The MGA tries to avoid server overloading by having a tree to handle these requests, yet a communication overhead is encountered in propagating address requests and replies between the requesting process and the MGA node and between the MGA nodes themselves.

The last measure we discuss is the probability of port blocking. Port blocking denies a process from joining a multicast group if the multicast port is occupied by another process in the host. Many approaches to this problem try to minimize the probability of blocking rather than eliminating it. The probability of port blocking is a good measure of performance for these approaches.

3.5 Multicast Address Management for Local Sessions

In many situations, all the group members and senders of the multicast session exist within the same domain. For this type of multicasting, requiring a unique address over the global Internet is just wasting of resources and efforts, since it is sufficient to make sure that the address is unique within the domain where the members exist. Adopting such assumption has two effects. First, it simplifies the address management task (a technique that cannot be used in the Internet because of its huge size can be used in a single domain). Second, the same address can be safely used in different domains at the same time, which greatly increases the number of multicast addresses available for local sessions.

For this approach to work, local multicast traffic can neither be forwarded outside its domain, nor can it be received from outside. Making such restrictions prevents any interference from other local multicast sessions running in other domains. Remaining, however, is the interference with global multicast sessions (group members and/or senders span the global Internet). To solve this problem, we divide the multicast address space into two partitions—one for each session type. The sizes of the two partitions should reflect the usage of each type of sessions (local or global) relative to the other.

Many techniques can be adopted for address management for local sessions. In fact, different domains can apply whatever techniques suitable for their uses. These techniques can be classified into two types: address collision avoidance and address collision detection and recovery. On the one hand, address collision avoidance techniques do not assign duplicate addresses to different groups at all. Detection and recovery techniques, on the other hand, assign addresses, not necessarily uniquely. They, then, detect collisions and recover from them when they occur.

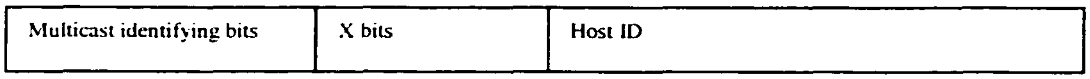

Fig. 4. Multicast addresses for local sessions.

3.5.1 Avoidance techniques

The first alternative is the use of a centralized server to manage multicast addresses within the domain. An entity requiring a multicast address sends a request to the server, which assigns it a free address if there is one available, otherwise the request is denied. We call this entity the multicast session initiator (MSI). MSIs that are assigned addresses are required to submit a periodic keep-alive report to the server. Upon termination of the multicast session, the MSI sends multicast session termination report. The server can reclaim an allocated address if it receives a termination report or if it does not receive the keep-alive report from its MSI. While this technique may be considered impractical for global sessions recognizing the size of the Internet and the number of sessions that may be running, it seems a reasonable choice especially for small-sized networks. Using a centralized server ensures uniqueness in address assignments, since the server itself has full knowledge of all running multicast sessions within the domain. Address blocking is minimal (as the server controls all the address space) and arises only when exhausting the address space assigned for the local sessions. However, address acquisition involves communication between the requesting entities and the server, thus the latency in getting the address and the communication and processing overhead may be high depending on the load placed on the network and in the server.

Another alternative is to use a fully distributed technique by partitioning the address space over some (or all) of the domain's hosts and letting those hosts capable of starting multicast sessions. The host ID (as part of the host IP address) is used within the multicast address generated by the host. Fig. 4 shows the format of such an address. The *X* bits shown in the figure are variable bits and distinguish between different addresses assigned by the host. This mechanism can be implemented by setting a process for multicast address management in the host. A process requiring an address should direct its request to the manager process. The technique ensures uniqueness in address assignments provided that there is no overlap between partitions given to hosts and provided that a host does not assign two address having the same *X* bits. Address acquisition is done within the host itself, minimizing the latency and processing overhead and there is no communication overhead. Address blocking, however, is relatively high— A host may exhaust its address space suffering address blocking while many other hosts may still have free addresses.

3.5.2 Detection and recovery techniques

In detection and recovery techniques, addresses are assigned with no regard to the running multicast sessions. Collision then is detected and resolved. One technique is to let each MSI assign itself an address (out of the entire address space of the local sessions) and register this address in a central registry. If the registry detects a collision (two or more MSIs using the same address), it signals all processes of the collided address (except only one) to abort the multicast session and start allover selecting different addresses. MSIs are required to send termination reports upon the end their multicast sessions. Similar to the centralized server discussed above, they are required to send periodic keep-alive reports to the registry as long as the multicast session is active. The registry deletes its entry of a multicast address when it receives a termination report or when it does not receive the periodic keep-alive reports. The decision which session should continue can be based on many criteria. Examples include the earliest session, the one with the large number of members, or the highest priority (in case sessions are prioritized).

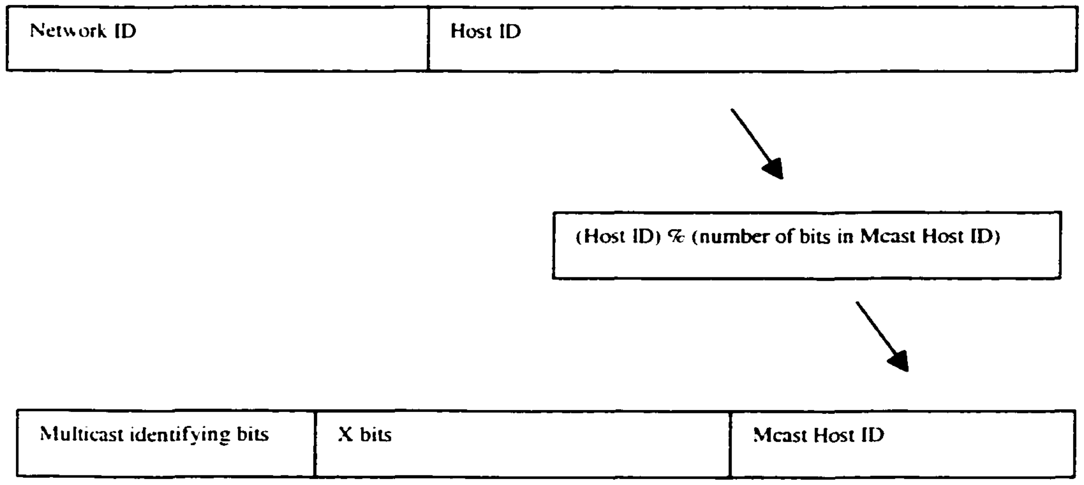
Network ID
Host ID
(Host ID) % (number of bits in Mcast Host ID)
Multicast identifying bits
X bits
Mcast Host ID

Fig. 5. Address calculation based on Host ID.

In order to minimize collision, processes calculate multicast addresses as a function of their host ID. Hashing can be used in case the number of bits reserved to distinguish hosts in the multicast address is smaller than those in the host ID. Fig. 5 gives an example of such hashing function. The *X* bits in the figure are variable bits that can be assigned by the operating system in a serial fashion for different multicast addresses initiated from within the host. The basic assumption in this technique is that the probability of collision is minimal, and it can be even minimized using an appropriate hashing function. The technique minimizes acquisition latency, blocking, and overhead. Things are different, however, in case of collision. In such a case, latency and processing overhead are high.

3.6 Multicast Address Management for Global Sessions

Many of the multicast sessions do not restrict their members to exist within a single network, rather, members are scattered over the entire Internet. For this type of multicast, selecting a unique multicast address globally is required to prevent cross talk. Techniques that may work reasonably well in local sessions may not perform the same in global ones. For example, using a central server to manage address assignment may not be practical because of the size of the Internet. Also, using an address collision detection and recovery technique does not seem attractive—asking a local multicast session to abort is much different from asking a global one to restart allover again.

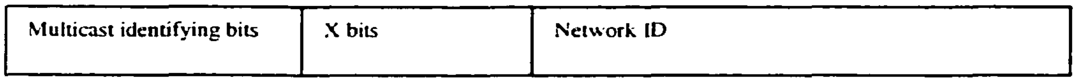

Fig. 6. Multicast addresses for global sessions in different network classes.

We present an address space partitioning technique that divides the multicast address space (the space assigned for global sessions) over the different domains to avoid address collision. The address block size of each domain depends on the domain size—larger domains take larger portions. Every domain can use whatever suitable technique to manage its global address space. One technique is to have a central server within each domain to manage the assignment of addresses in a way similar to the central server in local address assignment. The use of a central server has the same advantages and disadvantages as introduced in local address assignment (low address blocking probability as an advantage and high address acquisition latency and high processing and communication overhead as disadvantages). Another technique is to further partition the address space assigned to the domain among the domain's hosts. Each host then manages its space. A host can calculate a global multicast address as given in Fig. 6. The X bits shown in the figure are variable bits assigned by the host differently for different multicast sessions. This technique reduces latency and overhead. Address blocking, however, is higher than using a central server.

3.7 Multicast Port Resolution

A multicast port number is an integral part in the multicast identity. For a process to receive a multicast traffic, it must join the port number of the group. If the port is not free, we say that there is a port-blocking problem, in which the process cannot join the group until the port is freed. To solve this problem, an agreement between the potential multicast members on the port number must be reached. In this section, we investigate the blocking probability experimentally. We also introduce an analytical model for the problem. Finally, we introduce a solution for it.

3.7.1 Experimental Work

In the following, we present the experimental work conducted in the study of the multicast port blocking problem. The experiments were conducted on 50 Sun workstations that range from Sparc 5 to Ultra 60 with Solaris 2.6 as the operating system. All workstations are connected via LAN. All workstations are general-purpose computers and are open for use by students, faculty, and staff.

The first set of experiments measures the inter-arrival time and the service time for UDP port requests. These two measures are used in the analytical model as will be explained in the next section. We implemented a simple program to test every port in the host periodically to determine if the port is free or is allocated. The period between any two consecutive tests is chosen to be small enough so that, for any port, only an allocation or a release, and not the two combined, can happen. The program keeps a table of port request events that has three entries: port number, start time of allocation (the first time a free port is discovered to be allocated), and end time of the allocation (the first time a previously allocated port is discovered to be freed). We ran the program for 24 hours on all 50 workstations.

TABLE 3 gives the results for the experiments. The average service time is the average time ports are found occupied by processes. The system average arrival rate is calculated as the summation of the number of port request arrivals in all hosts divided by the number of hosts, while the port average arrival rate is calculated as the system average arrival rate divided by the number of ports in the host.

TABLE 3

UDP PORT USAGE

Parameter	Value
Average service time	64.2269 minutes
System average arrival rate	0.0186486 per minute per host
Port average arrival rate	$2.890718 * 10^{-7}$ per minute per port

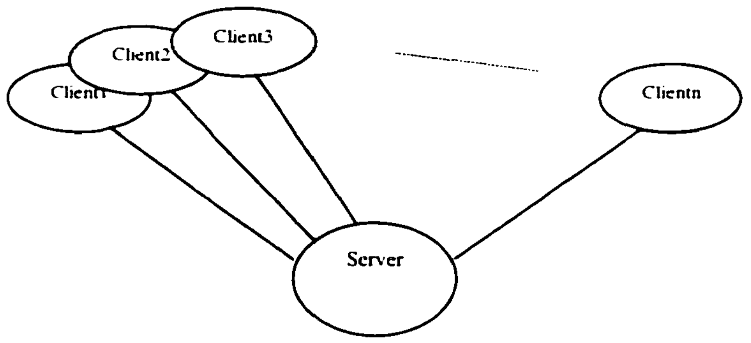

Fig. 7. Client server architecture of the port blocking experiments.

The second set of experiments investigates the probability of port blocking. We used a simple client/server technique to conduct the experiments (see Fig. 7). Initially, a server is started in a well-known machine and port number, and a number of clients are started in the rest of the machines. We used a Unix shell script to start all clients in their machines. The server waits for connections from the clients. When all clients connected, the server generates a port number at random and sends it to all clients. The server then waits to get the results back from all clients. A client, on the other hand, connects to the server, reads a port number from the server, tests to see if the port is available, and sends the result of the test back to the server.

This test tries to determine the availability of a given port at all clients' machines at one moment of time. Since the server sequentially unicasts its messages to the clients, there is a difference in time between receiving the message by different clients. This time difference compared to port requests' inter-arrival time and service time, however, is so negligible that no activities (requests or releases of ports) are expected to occur during it. This test mimics a multicast group formation, in which there are number of clients trying to allocate the same port for the multicast. If one or more clients fail to allocate the port, the group formation fails. The process is repeated 10000 times for 5 groups of sizes ranging from 10 to 50. Fig. 8 gives the blocking probability results of the experiment. The x-axis is the group size and the y-axis is the blocking probability, which is the number of group formation failures divided by the number of trials (1000). From the figure, for a group of size 10, there are 2 group formation failures out of 10000 trials. Also, as the chart shows, increasing the group size increases the chances of group formation failures. This is an expected result, since there are more processes to agree on the same port.

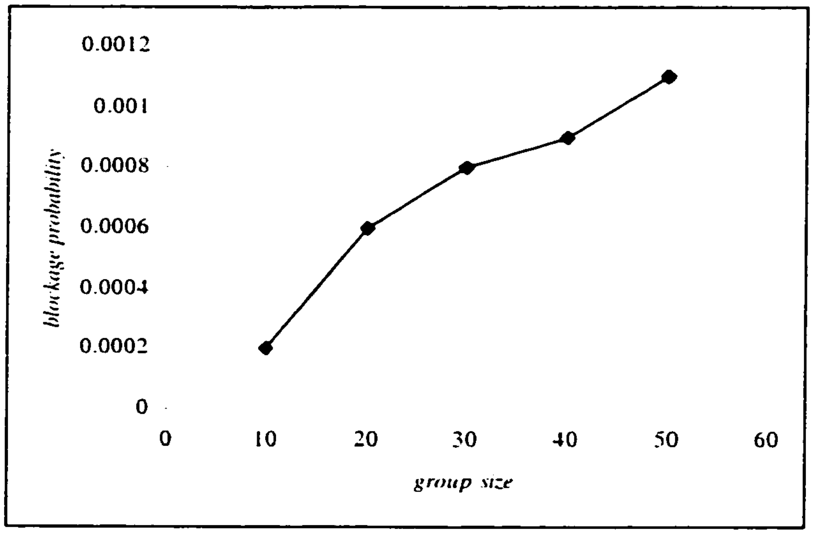

Fig. 8. Port blocking probability as a result from the experimental work.

3.7.2 Analytical Model

To model the multicast port blocking problem, we assume a system that consists of S hosts working independently. Each host manages N communication ports numbered from 1 to N. A port can be allocated for a variable duration of time T to one process at a time—

the possibility that a port can be allocated to more than a process at a time is excluded. When a process within a host requests a communication port, the operating system of that host gives it a port if there is one free. That port is labeled busy during the duration the process is using that port and cannot be given to another process. When the process releases the port, the port can be allocated to another requesting process. A process may request a specific port number from its host, or it may request any free one. In the latter case, the host's operating system gives ports to applications sequentially (starting from 1 towards N, then wraps around), that is when an application does not specify a port number for its port request, the application is given the next available port in the sequential order. To form a multicast group of S processes, one process, the initiator of the group, gets a free port in its host. To join the group, the other S-1 processes must obtain the same port selected by the first process.

Based on the above discussion, ports can be considered as servers with no waiting rooms (no queues) that receive requests at a specific rate and service each request for a specific duration of time. We assume that the requests for ports arrive in a Poisson distribution at a rate of λ requests/host/minute and the service time is distributed exponentially with mean $1/\mu$ minute. Since there are no queues for the servers (ports), this system can be considered as an **M/M/N/N** system, where:

- the first **M**: the arrival process is Poisson,
- the second **M**: the service time is exponential,
- the first **N**: is the number of servers (ports), and
- the last **N**: is the maximum number of requests.

The mean number of allocated ports (γ) can be given by the following equation [49], [74]:

$$\gamma = \frac{\lambda}{\mu}(1 - P_b) \qquad (1)$$

Where P_b is the probability of blocking.

To simplify the analysis, the blocking probability P_b can be neglected compared to 1, and the number of allocated ports (γ) can be approximated by:

$$\gamma = \frac{\lambda}{\mu} \qquad (2)$$

The probability of having any port free in one host can now be given by the equation:

$$P_1 = \frac{N-\gamma}{N} \qquad (3)$$

Where N is the number of ports.

Assuming that there are S hosts trying to join the same multicast port, the probability that all succeed to join the port is

$$P_2 = P_1^S \qquad (4)$$

The multicast port blocking probability can now be given as:

$$P_b = 1 - P_2 \qquad (5)$$

Substituting equations (3) and (4) into equations (5) gives the blocking probability as follows:

$$P_b = 1 - \left(\frac{N-\gamma}{N}\right)^S \qquad (6)$$

Fig. 9 gives the blocking probability (P_b) versus the group size (S) using the arrival rate and service time as resulted from the experimental work introduced earlier, and assuming that the number of dynamic ports per host is 64512, which is typical in many platforms. As the chart shows, 2 group formation failures may result out of 10000 trials. Increasing the group size increases the chances of failures.

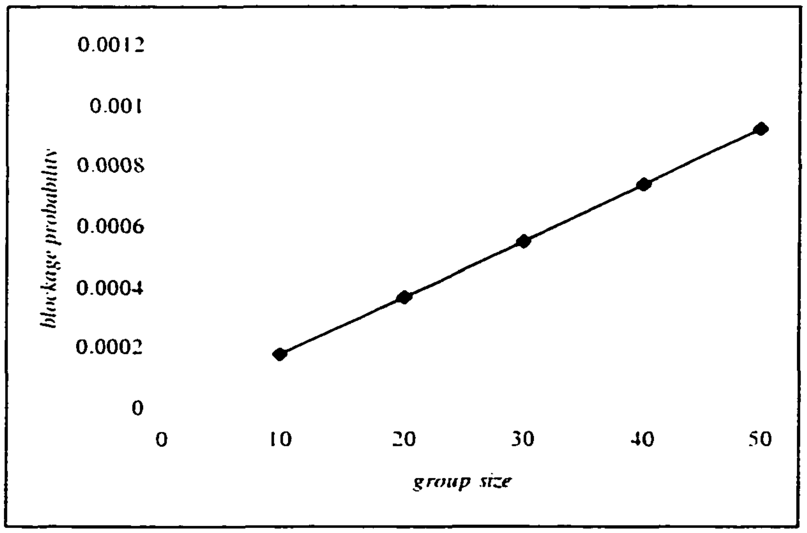

Fig. 9. Port blocking probability as resulted from the analytical model.

3.7.3 Multicast Port Resolution: A New Technique

As our experimental and analytical study show, multicast port blocking is an important aspect in multicast address management and connection control and the overall success of the collaboration session. One solution for the multicast port resolution problem is to have a central server that is responsible for negotiating the multicast port number between potential members. This approach, however, may be difficult recognizing the fact that port assignment within different hosts is a dynamic process: a free port now may not be so just after the negotiation phase has started. Also, negotiation with a central server suffers from many defects; examples are latency, server overload, and single point of failure. Another approach is to use virtual ports as discussed earlier [64]. This approach however requires system support and suffers from a large overhead in maintaining the map table.

In our new scheme, we suggest reserving a number of ports for multicast use only. This suggestion is justified recognizing the rapid use of multicast applications. Having a number of ports reserved for multicast use removes the competition of UDP in using these ports, which is the main reason for port blocking. To minimize the competition from other multicast sessions, we calculate the port number as function of the multicast address (A simple hashing function can be used for this purpose). Any process wishes to join the multicast group can calculate the port number knowing the multicast address. This way, we remove the need for a central server and its complication, and we also remove the overhead encountered in the virtual port approach. This technique still suffers from a port blocking in case there is collision in the hashing function. The probability of such collision, however, can be minimized by reserving a larger number of ports for multicast use.

3.8 Performance Evaluation

As introduced earlier, mainly there are four performance measures for multicast address management and port resolution: probability of address blocking, address acquisition latency, communication and processing overhead, and probability of port blocking. In the following, we study the performance of each multicast address management technique along these measures.

3.8.1 Address Blocking Probability

Following a previous study introduced by [64], we calculate the probability of address blocking in the various multicast address management techniques introduced earlier. A multicast address can be considered as a resource that can be requested by many groups but cannot be granted to two groups at the same time. Thus, a multicast address can be modeled as a server with a mean arrival rate λ: the multicast address request rate, and a mean service rate μ: the reciprocal of the mean multicast session duration during which the address is occupied.

A multicast address manager can be modeled using the Kendall notation as an *M/M/K/K* system—requests' inter-arrival and service time distributions are exponential, number of servers is K, and number of buffers is the same as number of servers [39], [74]. Based on this model, the blocking probability of multicast addresses can be given as:

$$P_a = \frac{\rho / k!}{\sum_{i=0}^{k} \rho^i / i!} \qquad (7)$$

Where: ρ is the traffic intensity and is given as: $\rho = \lambda / \mu$

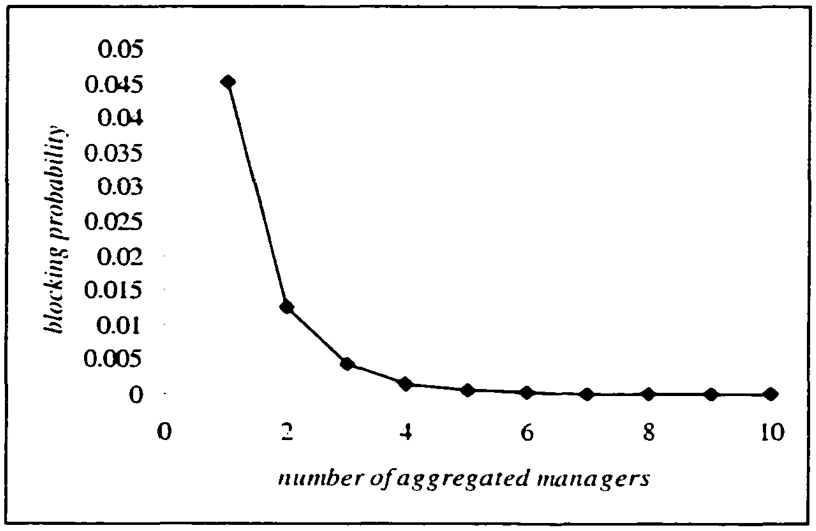

Fig. 10. Blocking probability when aggregating managers.

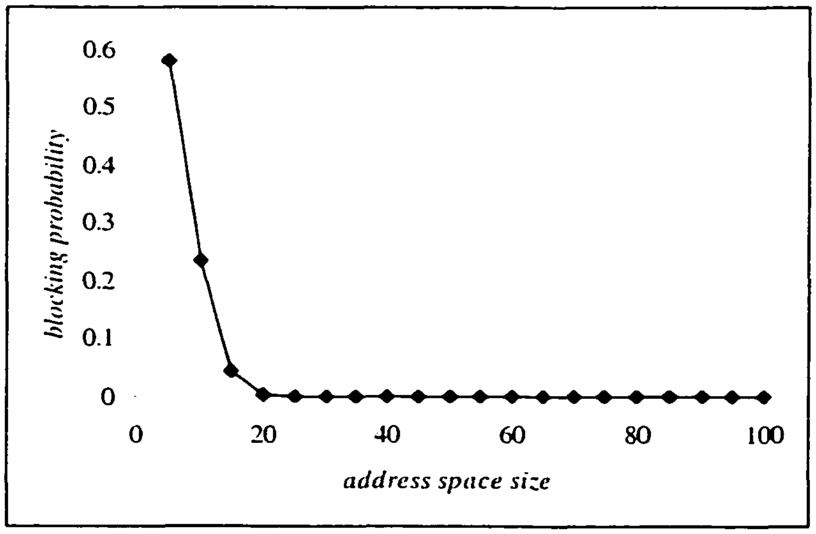

Fig. 11. Blocking probability when increasing the size of the address space.

Assuming a request rate of 1 request/day/host and an average session duration of 75 minutes, which is typical for many multicast sessions such as teleconferencing and distance learning, Fig. 10 and Fig. 11 show the blocking probability when aggregating managers and increasing the address space size. As the figures show, blocking probability decreases significantly by aggregating several managers into one and by increasing the address space size. The following is a study the blocking probability for MGA, address space partitioning over networks, the extended address, and partitioning over local and global sessions.

In MGA, there is a central server (organized in a tree) that manages multicast addresses. Since the central server is the maximum possible aggregation of managers, MGA has the minimum blocking.

The address space partitioning over networks is a semi-distributed management technique as it distributes the management task over networks, with the small network is assigned a small block of addresses. This is in effect increases the blocking probability as the above two figures indicate.

The extended address scheme lets every host manages its address space. The address space per host is very large, which results in a very small blocking. However, collision may result since routing is based on the 4-byte IP address only.

In partitioning over local and global sessions, the blocking probability for local sessions (whether using central server, partitioning the address space over the network's hosts, or using a detection and recovery technique) is negligible as a direct result of the large address space. As for global sessions, since every network is assigned a block of addresses taken from the global sessions' address space, the blocking probability is high, especially for small networks that are assigned small address blocks. The blocking probability gets even higher when partitioning the address space assigned for the network over the network's hosts.

3.8.2 Address Acquisition Latency

Address acquisition latency can be given by the following simple equation:

Acquisition latency = Processing time + Communication time

The processing time is the time required to decide on an address by the management entity, and communication time is the time consumed in message exchange between the address requesting and the management entities. In the following we study the acquisition delay for MGA, address space partitioning over networks, the extended address, and partitioning over local and global sessions.

The latency in MGA is very high due to the communication time required in exchanging requests and replies for multicast addresses between the requesting entity and the address manager.

In address space partitioning over networks, since there is a manager per network that manages multicast addresses within the network, the latency is high due to the communication time required. However, the latency is smaller than that experienced by MGA as messages are exchanged within the same network.

In the extended address scheme, there is no communication time and the processing time is negligible, which results in a minimal latency.

The last scheme to be discussed is partitioning over local and global sessions. For local sessions, partitioning the address over the network's hosts yields a minimal latency since the management process is performed within the host with no communication to an outside authority required. Using a detection and recovery technique also has a minimal latency as a process selects its address and starts the multicast session before communicating with the detection entity. Using a central server, however, results in a large latency due the required communication with the server. For global sessions, using a central server within the network to manage address assignment has a high latency, whereas partitioning the address space over hosts and letting them manage it has a minimal latency.

3.8.3 Communication and Processing Overhead

Mainly, there are two sources of overhead: communication and processing overhead. In the following, we discuss these overheads in MGA, address space partitioning over networks, the extended address, and partitioning over local and global sessions.

In MGA, Having a central server may overload the processing units. Also, it may lead to bottlenecks on the links leading to the server.

In the address space partitioning over networks scheme, the load is distributed among the individual managers (one per network), thus decreasing the processing and communication overheads per manager.

There is no communication involved in the extended address technique. Also, processing effort is distributed on all hosts, making the load per host minimal.

In partitioning over local and global sessions, and for local sessions, the communication overhead in having a central server is high. Partitioning the address space over the network's hosts does not encounter any communication and the processing overhead is minimal. The detection and recovery technique experiences some communication overhead. It also may experience large processing overhead in the rare case of detecting a collision and a multicast session requires a restart. For global sessions, having a central server suffers from a large communication overhead, while there is no such overhead in partitioning the address space over the network's hosts. TABLE 4 summarizes the performance of the various address management techniques.

TABLE 4

PERFORMANCE OF ADDRESS MANAGEMENT TECHNIQUES

Performance measure	MGA	Address partitioning over networks	Extended address	Local and global sessions
Address blocking probability	Minimum	Large	Negligible, results in address collision	Negligible for local sessions, high for global sessions
Address acquisition delay	Very high	High	Negligible	Vary for different techniques
Communication and processing overhead	High - results in bottlenecks	Small	Minimum	Vary for different techniques

3.8.4 Port Blocking Probability

Using virtual ports, the only way to have blocking is when all ports are occupied. Practically, the chance of such situation is negligible, as our experimental study has indicated.

Using Port Reservation, and as our experimental study has shown, most of the port requests are for unicast communication and the arrival rate for multicast port requests is almost zero. Thus, reserving even a small portion of the port space for multicast communications virtually eliminates the chances of port blocking based on equation (6).

3.9 Conclusion

In this chapter we have discussed multicast address management and port resolution in large inter-networks and, as an example, we have considered the Internet. We have presented previous work and have introduced new techniques for these two problems. We have evaluated the performance for our new techniques and compared them against previous ones.

First, for Multicast address management, we have noticed that multicast sessions can be classified into local sessions and global sessions and we have presented the idea of partitioning the multicast address space between these two session types. The size of each partition should reflect the usage of it relative to the other. If no such information is available, partitioning can be performed arbitrary between these two session types.

We have presented three alternatives for local multicast sessions: central servers (one per network), partitioning over network's hosts, and detection and recovery. All these techniques have a negligible blocking probability (as a direct result of the large size of the address space). However, partitioning the address space over the networks' hosts results in the lowest latency and communication and processing overhead.

For Global sessions, having a central server or using a detection and recovery approach may not be suitable because of latency and bottlenecks in the first technique and restarting a multicast session after detecting a collision may not appropriate in the second one. Partitioning the address space over the networks incurs large blocking probability but minimizes latency and avoids communication and processing bottlenecks.

We feel it is better to leave each network the choice of managing its address space, whether using a central server or partitioning the address space over its hosts.

Second, for port resolution, we have presented a technique that reserves a small portion of the host's port space to the multicast communication. Knowing that most of port requests are for unicast communication, the blocking probability is negligible. This technique does not eliminate the port blocking problem and it requires system support (to reserve part of the port space for multicast). However, it reduces the probability of port blocking to an insignificant level. Also, the system load is much smaller than that of having virtual ports (which also requires a system support) in maintaining the mapping table.

CHAPTER IV

MULTICAST ROUTING

This chapter introduces a multicast routing technique that is suitable for multimedia collaborative applications. Typically, the multicast groups for such applications are small and of slow membership dynamics. Moreover, the group members are not assumed to be located in a single domain; rather, they may span the global inter-network. An important requirement for multimedia collaborative applications is the delivery time—a shortest path delivery tree is of a major interest in constructing the multicast tree. Our routing technique constructs a distribution tree per source and it uses explicit membership messages to build and maintain the tree. Thus, it combines the advantages of both broadcast and prune and shared tree techniques; that is, it produces shortest path trees, yet the bandwidth consumed in building and maintaining such trees is minimal. We present a simulation study to compare the performance of our technique against broadcast and prune and shared tree techniques. Results of the simulation show that our technique outperforms previous ones.

4.1 Introduction

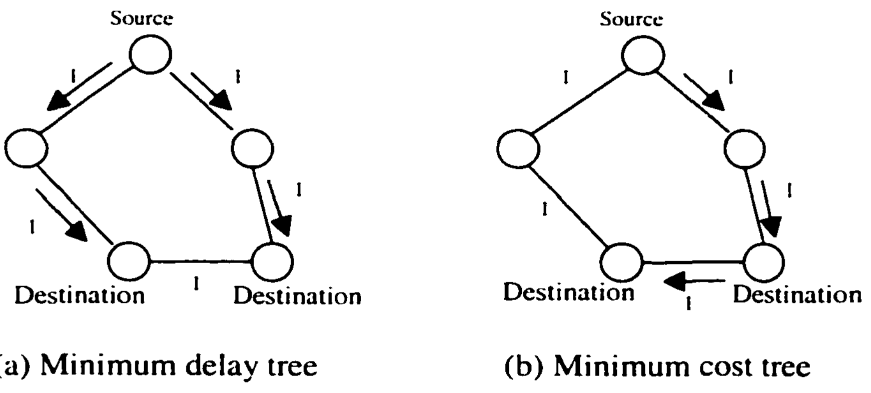

Fig. 12. Minimum delay vs. minimum cost routing.

Multicast routing concerns the construction of a tree—named a distribution tree—that is used to efficiently deliver data from a source to multiple destinations. Optimizing the distribution tree can be done with respect to either the cost or the delay of the tree. Building a minimum cost tree is the Steiner tree problem and is proven to be NP-complete [36], [42], [43]. Building a minimum delay tree corresponds to finding the shortest path tree. In contrast with the Steiner tree, the shortest path tree is a tractable problem with many existing efficient algorithms [22]. Fig. 12 contrasts the minimum cost and minimum delay for an example network that has all links of the same cost. Fig. 12-a gives a shortest path tree from a source to two destinations. The cost of the tree in this example is 4 units, while the number of links from the source to each destination is 2. Fig. 12-b gives the minimum cost tree from the source to the same destinations. The cost of the tree in this example is 3 units only (minimum cost) while the average number of links from the destinations is 2.5, resulting in a lager delay than the previous example. Two factors guide the research in multicast routing: advances in the communications technology and application requirements. The first factor lessens the importance of the tree cost, whereas the second suggests the need to have a minimal delay delivery. To this end, most of the existing multicast protocols sacrifice the cost in favor of the delay of the tree.

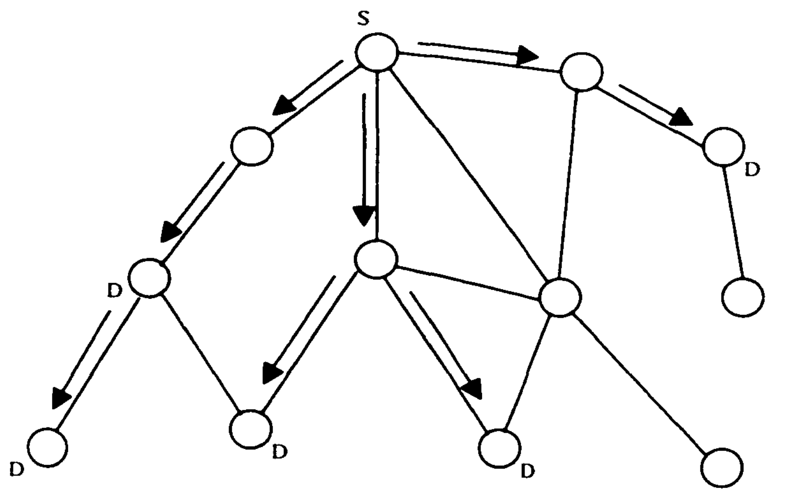

Fig. 13. Source rooted routing.

Multicast routing protocols mainly fall into one of two categories: broadcast and prune and shared tree routing. Broadcast and prune routing builds a tree per each source [23], [62]. Fig. 13 shows a source rooted routing tree. Broadcast and prune routing relies on broadcasting the multicast data packets periodically along the network edges. To determine the paths along which the multicast data are forwarded the reverse path forwarding is used [16]. On receiving the multicast traffic, uninterested routers prune themselves from the distribution tree. The periodic broadcast of multicast traffic is to keep the tree updated in response to group dynamics. Protocols based on this technique provide the shortest path tree from the source to the destinations. The main disadvantage, however, is the waste of the network bandwidth in building and maintaining the tree due to the periodic broadcast of multicast data. The problem gets much worse for large inter-networks where members are sparse. Also, per group/source information is stored in the routers to form the distribution tree, which imposes a scalability problem when there are many multicast groups and the number of sources per group is large. Examples of this technique include distance vector multicast routing protocol (DVMRP) [83], [78] and multicast extension to OSPF (MOSPF) [57].

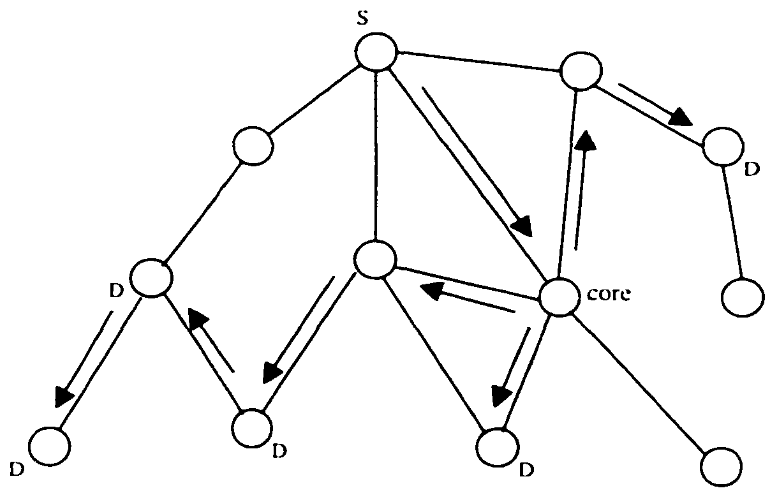

Fig. 14. Core rooted routing.

Shared tree routing relies on explicit membership messages originated from the destinations and directed towards a known core to build and maintain the distribution tree. Fig. 14 gives an example of the core rooted routing. Initially, a core is selected. When receiving membership messages from destinations, the network routers update their entries to build the multicast tree and forward the message to other routers along the way to the core. All multicast traffic destined to the group is unicasted from the source to the core, from where it follows the multicast tree to reach the group members. This technique eliminates the need to broadcast the multicast data to construct the tree, saving a valuable bandwidth, especially if the group members are sparsely located in a large network. Also, it stores per group information in the routers, making it more scalable compared to broadcast and prune. However, building a shared tree may not give the shortest path tree for some sources. Another disadvantage is the traffic concentration around the tree core, which may result in large delay and even packet loss. Last, but not least, selecting the core of the tree is a very difficult problem. Examples of this approach include core based tree (CBT) [8], [9] and protocol independent multicast-sparse mode (PIM-SM) [20].

From the above discussion, each technqiue has its strong and weak points. In this work, we introduce a new multicast routing techniuqe that combines the advantages of both approaches while avoiding their shortcomings to better serve multimedia collaborative applications. The new technique provides the shortest path tree, yet it avoids the large bandwidth consumed in building and maintaing the tree. Basic assumptions for our technique to work is small size and slow membership dynamics of the multicast groups, which are typical in multimedia collaborative applications. We demonstrate the effectiveness of our technique by a simulation study to compare its performance against broadcast and prune and shared tree techniques.

4.2 Multicast Routing: the Challange

In its general form, the multicast routing problem can be stated as the efficient construction of multicast distribution trees from sources to multiple destinations while maintaing the following assumptions:

1. Destinations' information is destributed within the network's routers.
2. Destinations' information is dynamic.
3. Sources' information is not kept at all within the network.

A challenge that faces this problem is the size of the network where sources and destinations are located, which is normally very large. Satisfying these three assumptions, a solution to the problem can be one of three:

1. Let the sources look for the destinations in the entire network.
2. Let the destinations look for the sources in the entire network.
3. Advertise a rendezvous points where sources and destinations can meet.

Letting the sources explore the entire network to find destinations and build the tree is the broadcast and prune technique described earlier. However, and as explained above, this technique is very costly in building and maintaing the tree, rendering it impractical in large networks, where the destinations are sparse.

Letting destinations explore the entire network to find sources is more costly than the previous one, since the solution requires each destination to broadcast membership messages along the network edges in searching for sources. Usually, the number of destinations is larger than the number of sources. Since sources are not known to the network, these messages should be delivered to every end system in the network, wasting not only the network bandwidth, but the computing power of the end systems as well.

The third solution; having rendezvous points where sources and destinations can meet; tries to get around the first and the third assumptions stated above. An example of this solution is core-based tree—a core is selected and destinations and sources send their membership information and data to the core. In this thesis, we further explore this third solution—having a rendezvous points where sources and destinations can meet—and we present a technique based on this idea.

4.3 Taxonomy

Multicast routing techniques can be classified along three diminisions. Depending on who initiate the tree construction, a multicast routing technique is a source-initiated or destination-initiated. According to the location of the tree root, it is a source-rooted or center rooted. Last, it can use the multicast data (data-driven) or explisit membership messages to build the tree.

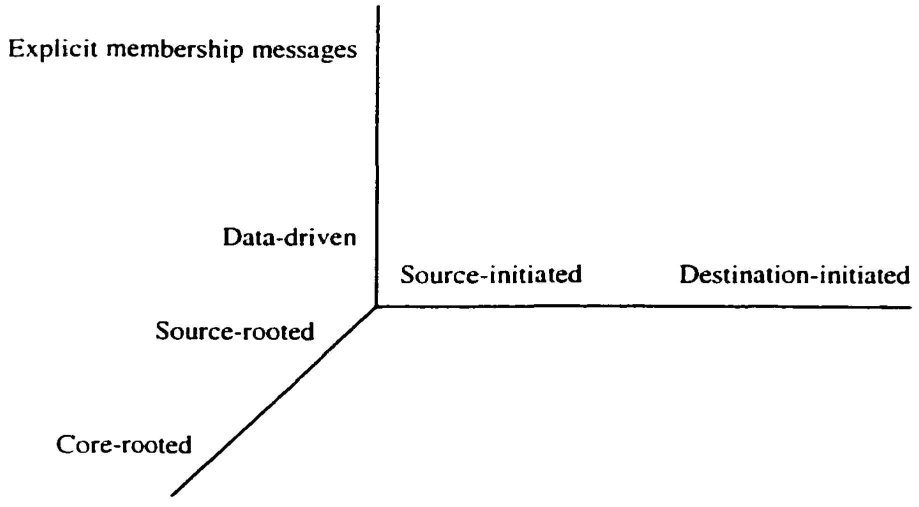

Fig. 15. Taxonomy of multicast routing techniques.

Fig. 15 shows the taxonomy of multicast routing techniques. Based on this taxonomy, broadcast and prune is a source-initiated, source-rooted, data driven routing, wheras core based tree is a destination-initiated, center-rooted, and uses explicit membership messages to buld the tree. Our new technqiue is hybrid (source/destination initiated), source-rooted, and uses explicit membership messages to build the tree. TABLE 5 summarises the taxonomy of multicast routing approaches.

TABLE 5

TAXONOMY OF MULTICAST ROUTING APPROACHES

Approach	Membership msg.	Root of tree	Tree initiator
Broadcast and prune	Data driven	Sources	Sources
Shared tree	Explicit	Selected core	Destinations
The new technique	Explicit	Sources	Destinations - sources

4.4 Performance Indexes

The performance of a multicast routing technique can be evaluated by several measurements. *Cost, delay, traffic concentration, join latency, overhead,* and *scalability* are the most popular measures. This is because they are easy to measure and they directly affect the members of the multicast group as well as the entire network performance. The *cost* of a multicast technique can be defined as the total network bandwidth consumed in delivering the multicast data to its destination. Finding the minimum cost distribution tree is an NP-complete problem. However, many heuristics have been introduced to find a low cost distribution tree [23], [36], [75].

A very important factor in multicast routing is the *average end-to-end delay* due to data delivery from sources to destinations. Many recent applications, such as video conferencing, impose minimum delay restrictions on the data delivery. Constructing the minimum delay tree usually conflicts with minimizing the cost of the tree. The increasing advances in the communications technology and the imposed application requirements, however, steer most of the research effort in this field towards minimizing the delay.

When building the multicast distribution tree, caution should be taken not to *concentrate traffic* around some routers in the network. Traffic concentrations may lead to delay, and even losses due to overloaded routers, in the delivery of the multicast data.

Join latency is the time experienced by an end system that has issued a join membership report to its router until the time it receives the multicast data. The goal is to minimize this latency.

Another important measure for multicast routing techniques is the *overhead of building and maintaining the distribution tree.* This overhead takes the form of network bandwidth, routing state stored and computational effort performed within the routers. The most important of these forms, however, is the network bandwidth overhead, since this affects the overall performance of the network. The goal is to build the tree with the minimum possible overhead.

Recently, Large groups, which are sparse and span a large inter-network, are common in many applications. When designing a routing protocol, *scalability* of the protocol is a very important factor to accommodate these groups. Scalability is a measure of how well a routing technique performs when the network size or the number of destinations and sources increase.

4.5 Multicast Routing: A New Technique

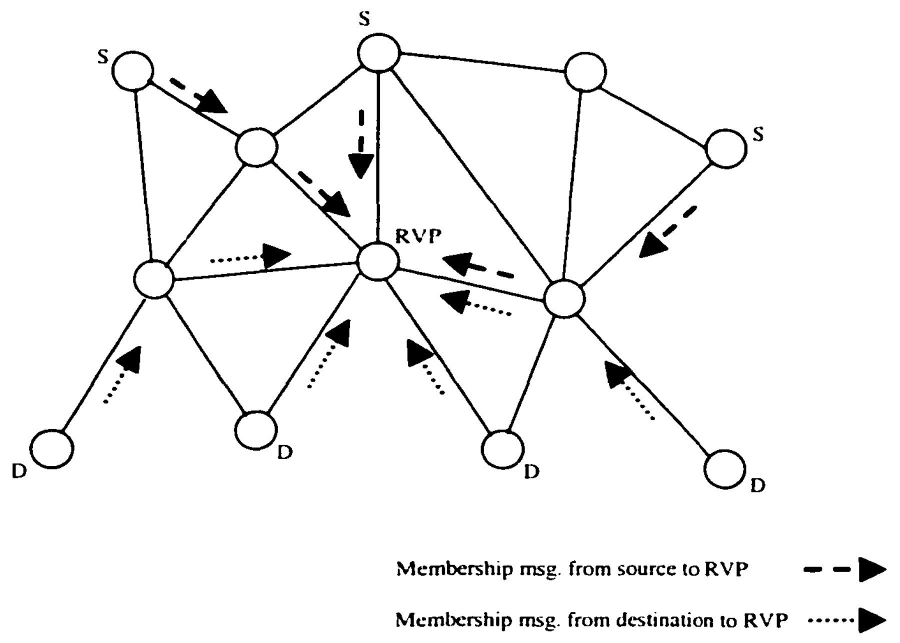
S
S
S
RVP
D
D
D
D
Membership msg. from source to RVP
Membership msg. from destination to RVP

Fig. 16. Membership messages from destinations and sources to the RVP.

In this section we present a new routing technique that is suitable for multimedia collaborative applications. Two basic assumptions underlie this technique: group members are few and membership dynamics is slow. The new technique is a source rooted—it constructs a delivery tree rooted at each source—and it uses explicit membership messages to construct the tree; that is, it avoids the periodic broadcast of data.

The basic idea behind our technique is to advertise a *rendezvous point* (RVP) where sources and destinations can meet. Two lists are kept in the RVP: the *destination list* (which contains the destinations' addresses) and the *source list* (which contains the sources' addresses). Beside the multicast group that is used to deliver data from sources to destinations (which we will call *data group*) another multicast group (the *source group*) is used to multicast control messages from the RVP to the sources of the multicast data group.

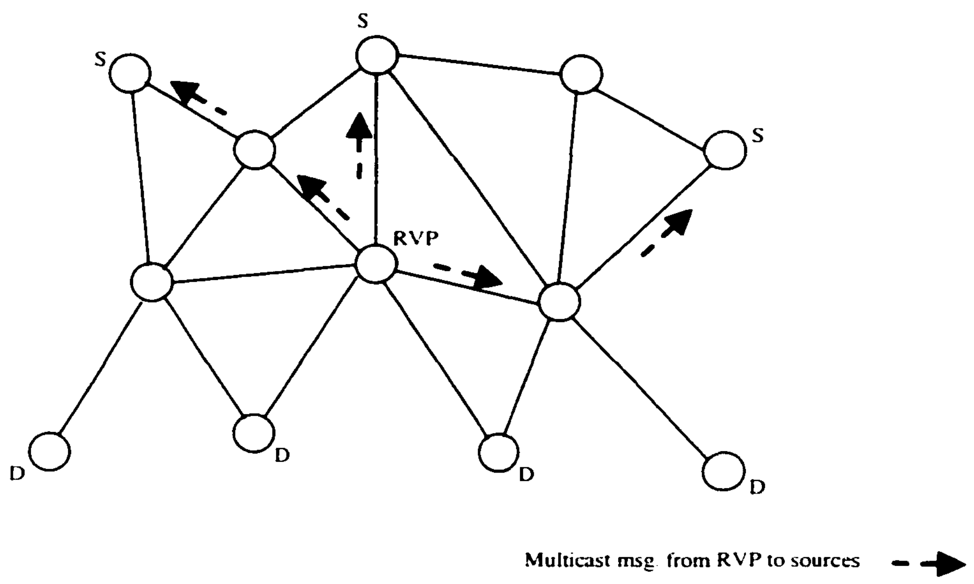

Fig. 17. Multicast message from RVP to sources.

An end system wishing to be a member of the data group is required to unicast a *destination membership join* message to the RVP before it can receive traffic destined to the group. Similarly, an end system wishing to multicast data to the data group (be a source of the group) is required to unicast a *source membership join* message to the RVP and become a member in the source group before it can send to the data group. A destination/source membership join message contains the message type (join the data/source group) and the address of the sender of the message. Likewise, a destination/source that wishes to quit the data/source unicasts *destination/source membership leave* message to the RVP. The RVP uses the destinations' and sources' membership messages to build and maintain the destination and source lists. Fig. 16 shows the membership messages sent from the destinations and sources to the RVP.

Periodically, the RVP multicasts the source group with two messages: *data group list* message and *source group list* message. These two messages contain the destination and the source lists. Also, on receiving a source membership join message, the RVP unicasts these two messages to the new source. Sources use these two messages to build and maintain their own lists, which are used to construct the delivery trees rooted at the sources. When a source receives a message of group list—be it a data group or source group list—from the RVP, it forms a *construct tree* message for the group and sends it to its router. The construct tree message contains the group address, the source address, and the member list of the group. Fig. 17 gives the messages sent from the RVP to the sources.

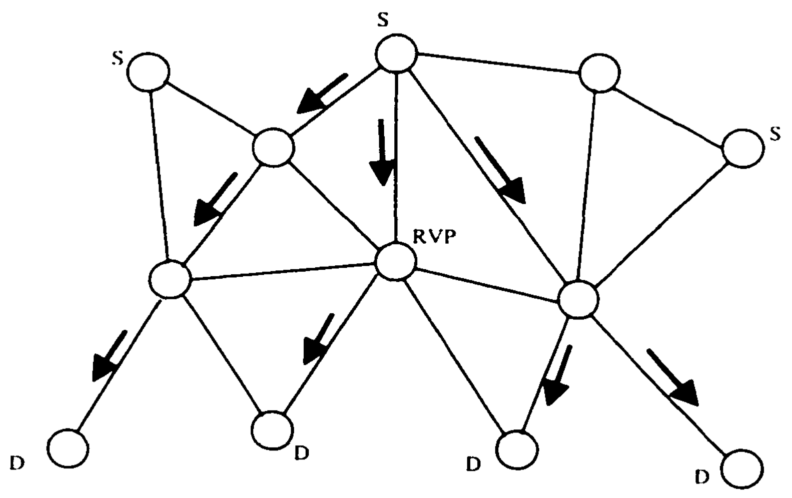

Fig. 18. Multicast tree for one source.

Each router along the paths of the distribution tree is required to maintain a table that determines the outgoing links for each multicast group/source pair. Entries of this table take the following format:

Group address	*Source address*	*List of outgoing links*

When a router receives a construct tree message, it extracts the group address, the source address and the group member list out of the message (routers do not differentiate between source and destination groups). For each group member, the router determines the outgoing link based on the shortest path from the source to that group member(Unicast routing table information can be used in deciding for the shortest path from the source to a specific destination). Group members sharing an outgoing link are grouped together along with the original source address and group address in another construct tree message. The router then forwards the construct tree messages along the calculated outgoing links to other routers. This process continues until the construct tree messages reach their destinations. Fig. 18 gives the resultant multicast tree for one source. Using the unicast routing table and assuming that this table does not produce any loops, grouping the destinations sharing the same outgoing link into new construct tree messages and forwarding the new messages along the calculated links guarantees a loop free routing tree. That is, our technique does not introduce any loops in the resultant tree and its success in this aspect depends on the unicast routing table used.

In order to capture group membership dynamics, the RVP keeps a timer for every entry in its lists. Every Destination as well as every source is required to periodically send a *renew membership* message to the RVP. On receiving the renew membership message, the RVP sets the timer for the destination/source. The RVP deletes those entries that have timed out from its lists. Similarly, routing table entries timeout after a certain period of time that is related to the frequency of the membership messages sent from destinations and sources. The purpose of these timeouts is to prevent stale entries within the RVP as well as within the routing tables resulting from destinations and sources that left their multicast groups. TABLE 6 summarizes the messages used in our routing technique.

TABLE 6

MESSAGES USED IN THE NEW ROUTING TECHNIQUE

Message type	Message source	Message destination	Description
Destination membership join	An end system wishing to join the data group	RVP	To build and maintain the data group list in the RVP
Destination membership leave	A member wishing to leave the data group	RVP	To build and maintain the data group list in the RVP
Source membership join	An end system wishing to send to the data group	RVP	To build and maintain the source group list in the RVP
Source membership leave	A source wishing to leave the source group	RVP	To build and maintain the data group list in the RVP
Data group list	RVP	Sources	To build and maintain the data group list in the sources
Source group list	RVP	Sources	To build and maintain the source group list in the sources
Construct tree	Sources and routers	Routers	To build the delivery tree rooted at a specific source
Destination renew membership	Data group member	RVP	To prevent stale entries in the data list in the RVP
Source renew membership	Source group member	RVP	To prevent stale entries in the source list in the RVP

This technique depends heavily on the performance of the RVP as a center that receives control messages from sources and destinations, maintain the lists, and forward the control messages back to the sources. To tackle the single point of failure within this technique, each source to the group is also required to maintain the two lists. In case of RVP failure, which can be detected by the absence of the periodic messages, one of the sources takes over the functionality of the RVP. In case there are multiple sources in the group, a simple resolution technique can be used to decide which source should take over as an RVP—the source with the largest address for example.

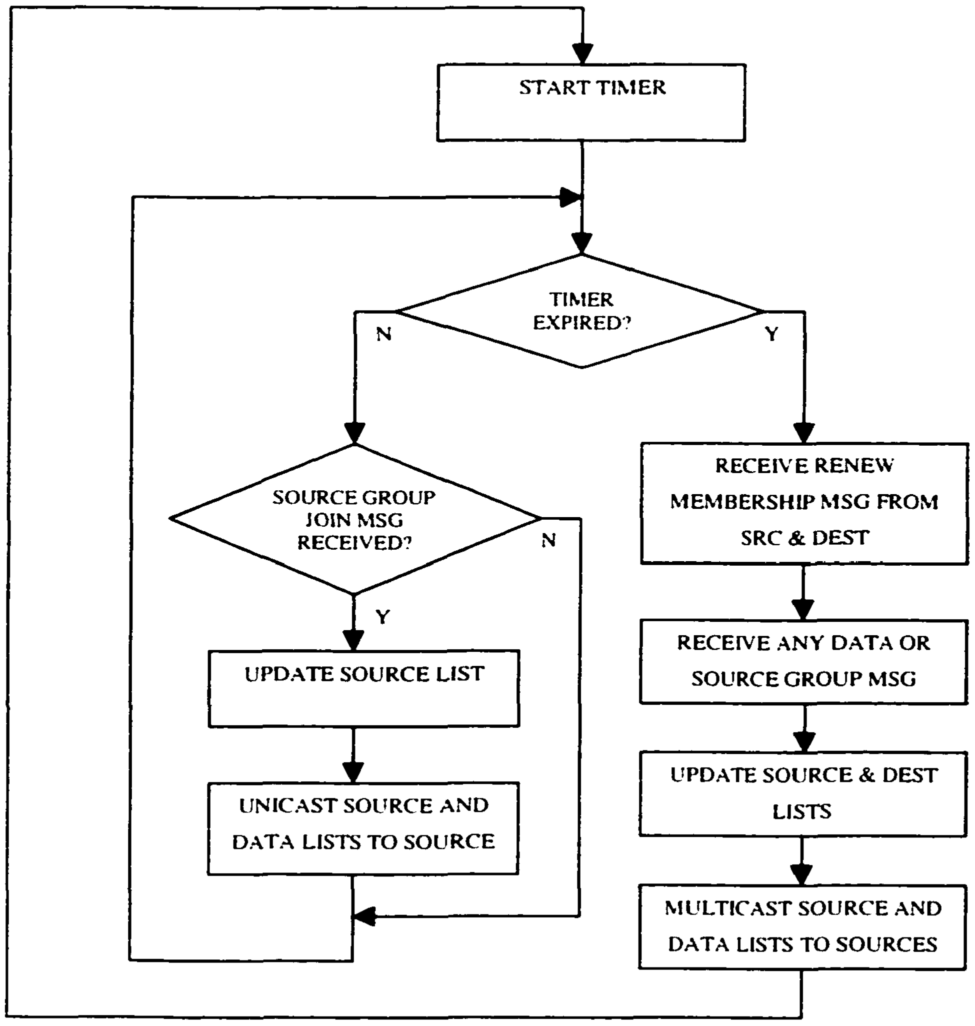
START TIMER
TIMER EXPIRED?
N
Y
SOURCE GROUP JOIN MSG RECEIVED?
N
Y
UPDATE SOURCE LIST
UNICAST SOURCE AND DATA LISTS TO SOURCE
RECEIVE RENEW MEMBERSHIP MSG FROM SRC & DEST
RECEIVE ANY DATA OR SOURCE GROUP MSG
UPDATE SOURCE & DEST LISTS
MULTICAST SOURCE AND DATA LISTS TO SOURCES

Fig. 19. RVP flow chart.

Remaining to this discussion is the selection of the RVP. The main objective is to select an RVP so as to minimize the overhead associated with sending the control messages from the destinations and the sources to the RVP and back from the RVP to the sources. Since we assume small group with slow membership dynamics, a good heuristic is to select one of the sources as the RVP[1].

[1] Selecting one of the sources as the RVP may reduce the communication overhead since the communication overhead of the selected source is avoided.

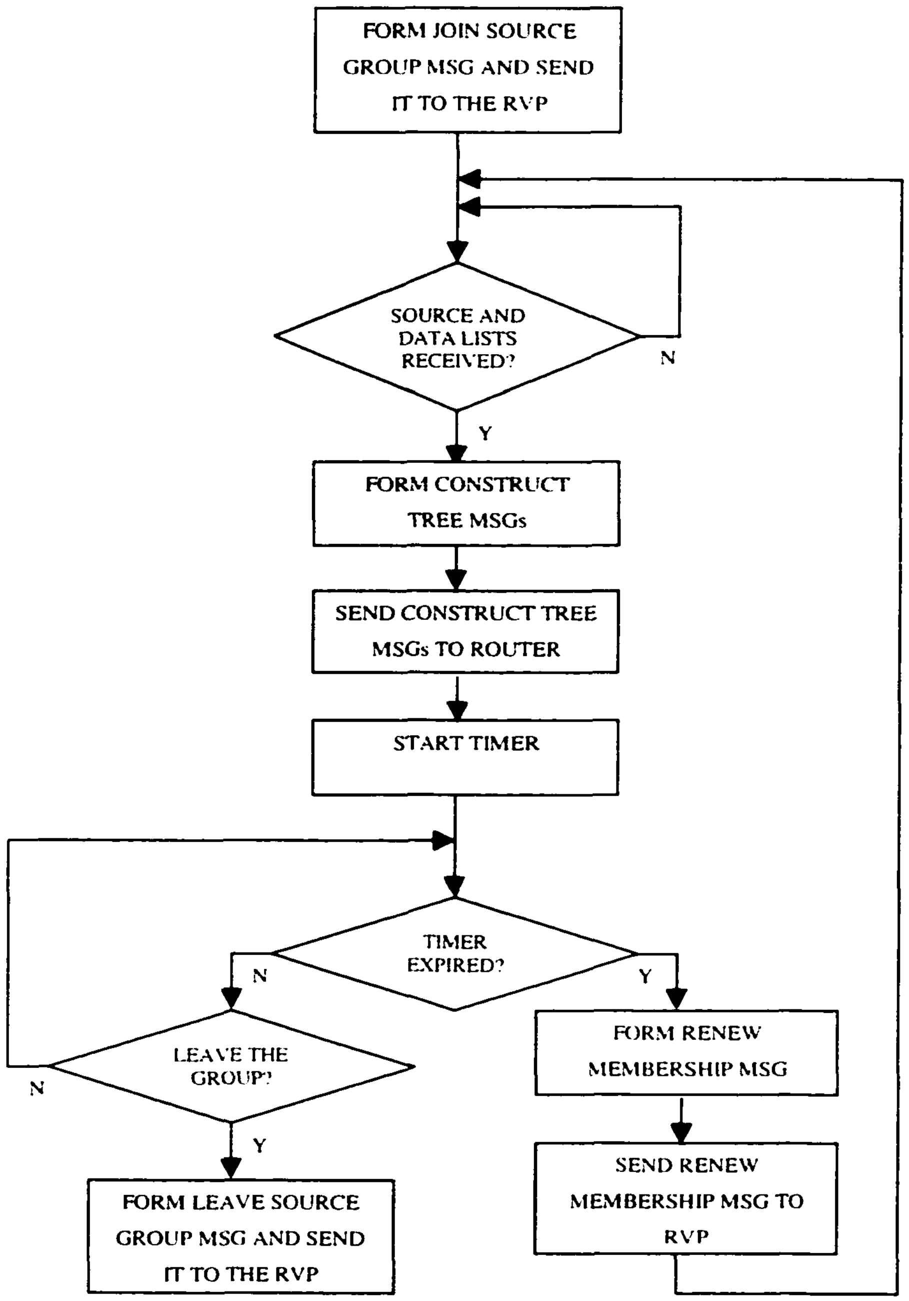
FORM JOIN SOURCE GROUP MSG AND SEND IT TO THE RVP
SOURCE AND DATA LISTS RECEIVED?
N
Y
FORM CONSTRUCT TREE MSGs
SEND CONSTRUCT TREE MSGs TO ROUTER
START TIMER
TIMER EXPIRED?
N
Y
LEAVE THE GROUP?
N
Y
FORM RENEW MEMBERSHIP MSG
SEND RENEW MEMBERSHIP MSG TO RVP
FORM LEAVE SOURCE GROUP MSG AND SEND IT TO THE RVP

Fig. 20. Source flow chart.

4.6 Modeling the New Technique

In this section we model the routing technique discussed in the previous section using flow charts. There are four entities to be considered: RVP, sources, destinations, and routers. Each of these entities interacts with the others. However, in order to simplify the discussion, we give the flow chart of each entity individually. The relationship between these entities can be inferred from the individual charts.

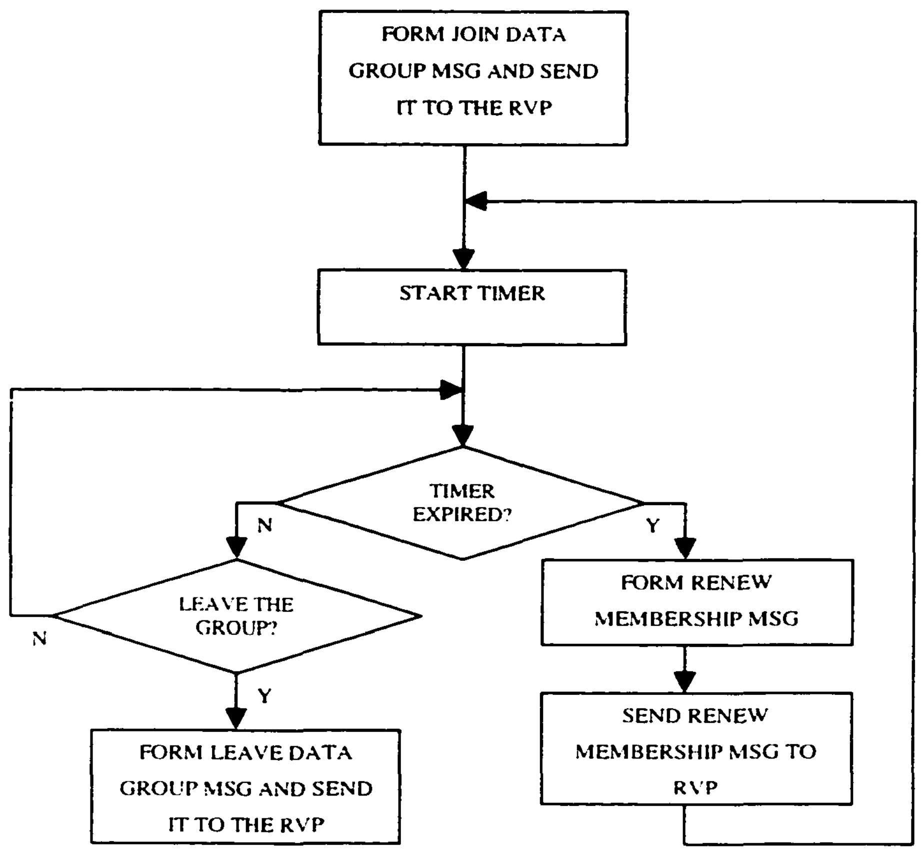
FORM JOIN DATA GROUP MSG AND SEND IT TO THE RVP
START TIMER
TIMER EXPIRED?
N
Y
LEAVE THE GROUP?
N
Y
FORM LEAVE DATA GROUP MSG AND SEND IT TO THE RVP
FORM RENEW MEMBERSHIP MSG
SEND RENEW MEMBERSHIP MSG TO RVP

Fig. 21. Destination flow chart.

4.6.1 RVP Flow Chart

In Fig. 19, we give the flow chart for the RVP. As Fig. 19 indicates, the RVP starts a timer. After the expiration of the timer, the RVP collects all information of source and data lists received through the source membership join/leave, source renew membership, destination membership join/leave, and destination renew membership. The RVP then updates its two lists based on the received information and multicasts the lists to all sources. On receiving a source membership join message, the RVP unicasts the new source the source and data lists.

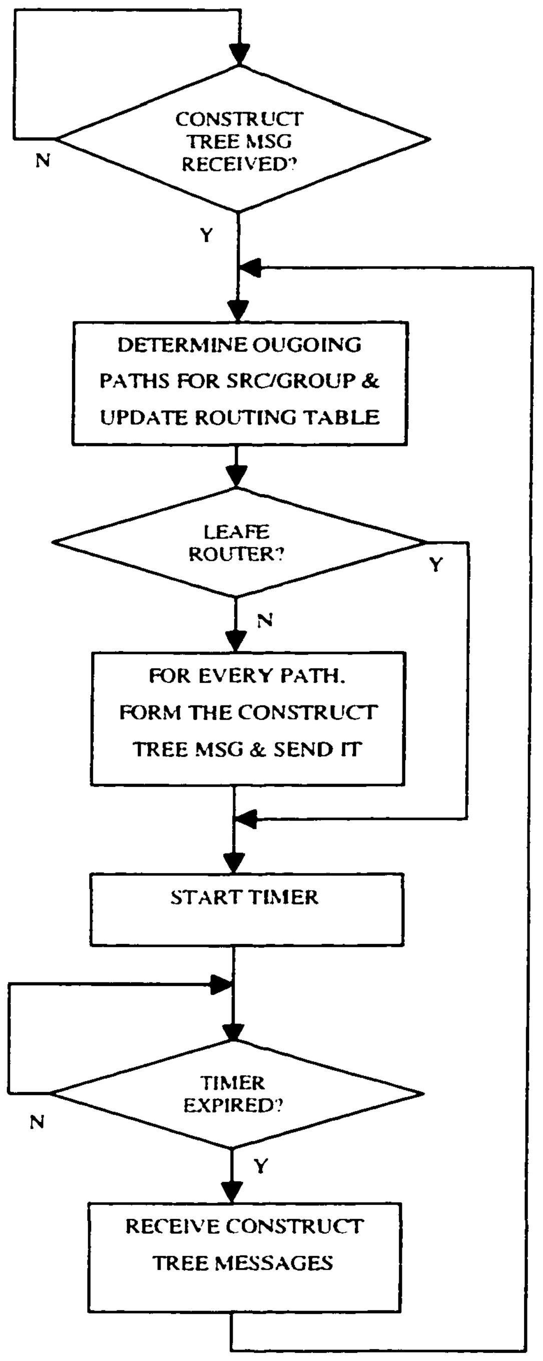
CONSTRUCT TREE MSG RECEIVED?
N
Y
DETERMINE OUGOING PATHS FOR SRC/GROUP & UPDATE ROUTING TABLE
LEAFE ROUTER?
Y
N
FOR EVERY PATH, FORM THE CONSTRUCT TREE MSG & SEND IT
START TIMER
TIMER EXPIRED?
N
Y
RECEIVE CONSTRUCT TREE MESSAGES

Fig. 22. Router flow chart.

4.6.2 Source Flow Chart

The flow chart of the source algorithm is given in Fig. 20. As Fig. 20 shows, the source first joins the source group by sending a source membership join message to the RVP. The source then waits for the source and data lists to be received from the RVP. On receiving the two lists, the source forms two construct tree messages, one for the source group and the other for the data group, and sends them to its router. The source then starts a timer and when the timer expires, it sends renew membership message to the RVP. When the source wants to quit the group, it sends a source membership leave message to the RVP.

4.6.3 Destination Flow Chart

As Fig. 21 indicates, the destination first forms a destination membership join message and sends it to the RVP. It then starts a timer and periodically sends a destination renew membership to the RVP. When the destination wants to quit, it sends a destination membership leave message to the RVP.

4.6.4 Router Flow Chart

The flow chart of the routers is given in Fig. 22. As Fig. 22 shows, the router waits for a construct tree message. On receiving the message, the router determines the outgoing shortest paths from the source to the group members and updates its routing table. If not a leaf router, it then forms construct tree messages for other routers and forwards the messages to the routers along the paths to the destinations. The router maintains a timer to prevent stale entries in its table.

4.7 Variations of the Routing Technique

In the previous sections we have introduced one variation of our routing technique. There are, however, three other variations of the technique introduced. The performance of each variation depends on the number of sources relative to the number of destinations. In the technique introduced above, the RVP maintains two lists: one for the destinations and the other for sources. In two variations, the RVP is required to maintain one list only. In both variations, destinations and sources still are required to send their membership messages towards the RVP. The RVP builds one list for either group (source or destination). A multicast tree, rooted at the RVP, is build for the other group using the membership message in a way similar to that used in CBT. In the third variation, the RVP does not maintain any list at all. These variations try to minimize the workload placed on the RVP by minimizing the number of lists maintained and the number of membership received by it.

If the number of destinations is less than the number of sources, then the RVP maintains a destination list. Destinations are required to send their membership messages to the RVP as before. The RVP uses these messages to build the destination list. A source wishing to send to the group forwards a source membership join message to the RVP. Using these messages, a multicast tree, rooted at the RVP, are built for the sources exactly like CBT. Any modification on the destination list is multicasted from the RVP to the sources. A source receiving a destination list builds the multicast from that source to the destinations.

In the second variation, the number of sources is less than the number of destinations. In this case, the RVP maintains a list for the sources. All sources are required to send their membership messages to the RVP, which in turn uses these messages to maintain the source list. A destination wishing to be a member in the group forwards a destination membership join message to the RVP and a multicast tree for the destinations is built exactly like CBT. Any modification to the source list is multicasted to the destinations. A destination receiving the source list send a membership message towards each source and a tree is build exactly like the way trees are built in CBT.

In the last variation, the RVP does not maintain any list. As before, an RVP is advertised. All destinations are required to forward their destination membership join messages towards the RVP and a multicast tree, rooted at the RVP, is built exactly the same was as in CBT. An end system wishing to multicast to the group reliably unicasts a source membership join message to the RVP. The RVP multicasts a *construct new tree* message that contains the new source identity to all destinations. All destinations forward membership messages towards the new source given in the construct new tree message and the delivery tree, rooted at the source, is constructed similar to the way trees are built in CBT. To keep the tree updated, group members and sources are required to periodically send their information towards the RVP. This technique eliminates the overhead placed on the RVP in maintaining the lists. It, however, suffers from large delay in building the multicast delivery tree and large join latency.

4.8 Performance Analysis

To evaluate the performance of our routing technique, we conducted a simulation study to compare the performance of the new technique against that of broadcast and prune and CBT. We model the network by a connected directed graph where routers are nodes in the graph while links between routers are edges between nodes. We generate flat random graphs using the Georgia Tech Inter-network Topology Models (GT-ITM) [86]. The graphs used in our simulation range in size from 100 to 1000 nodes. The average degree of the graphs is 5. Every link in the graph has a delay value, which is generated at random. Every node in the graph has a queue length that corresponds to the incoming traffic of the node.

The simulation program is written in C++ on Solaris and it uses the multiplicative linear congruential method to generate pseudo random numbers from which the exponential and uniform random variates required in the various simulation events are generated. For a discussion in random variates, the reader is referred to [49]. Measurements out of the simulation include: *average delay*, *cost* in terms of bandwidth used, *traffic concentration* in terms of average queue and maximum queue length, *bandwidth overhead* in building and maintaining the tree, *memory usage* due state information stored, and *join latency*.

Given a randomly generated graph, the simulation starts by selecting an initial set of destinations and sources. For broadcast and prune (BAP) and our new technique, the shortest path trees, rooted at the sources, to all destinations are built using Dijkstra algorithm [22]. For CBT, a core is selected at random, and the shortest path tree, rooted at the core, is built from the core to all destinations. The simulation continues by generating traffic from sources to destinations. We assume that the traffic generated is for audio and video applications.

Two sets of experiments have been conducted. The first measures the performance of the routing techniques against the network size while fixing the group size to 10 members, whereas the other measures it against the group size for a network size of 1000 routers. The following is a presentation of the results of the simulation.

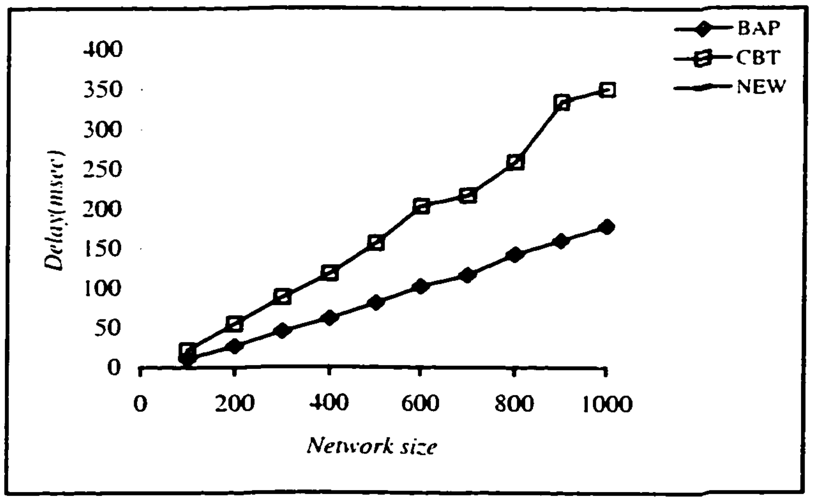

Fig. 23. Average delay vs. network size.

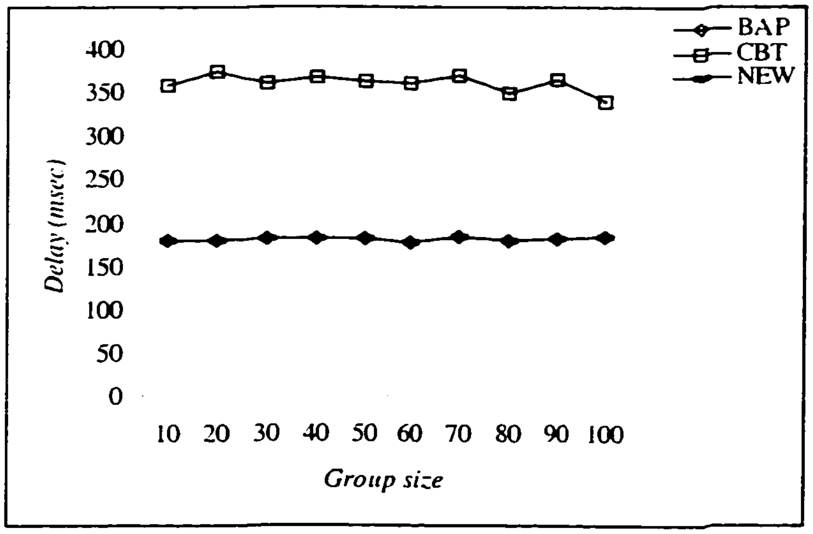

Fig. 24. Average delay vs. group size.

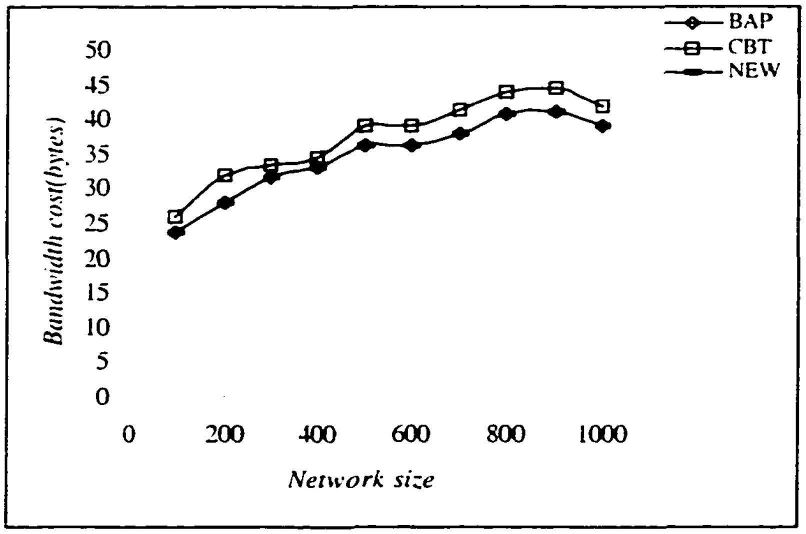

Fig. 25. Average bandwidth cost vs. network size.

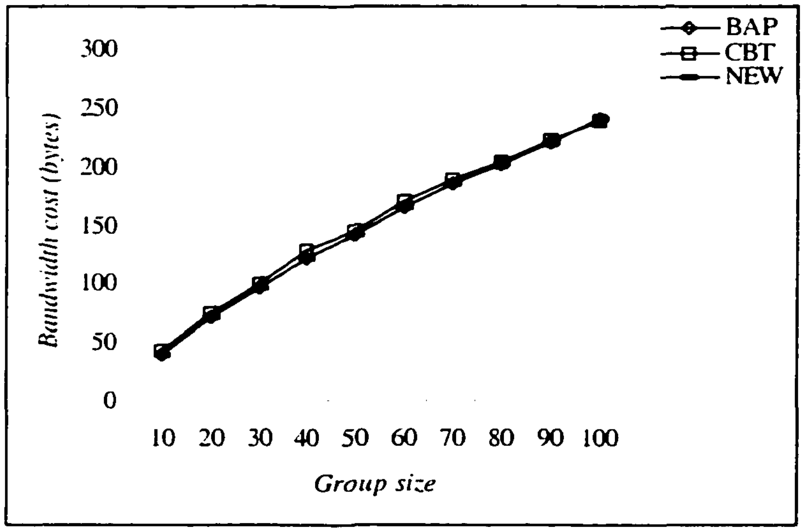

Fig. 26. Average bandwidth cost vs. group size.

The average delay of delivering the multicast traffic from the sources to the destinations is given in Fig. 23 and Fig. 24. As shown in Fig. 23 and Fig. 24, the delays for broadcast and prune and our new technique are almost the same. The delay, however, is larger for CBT. Since the shortest path tree is constructed in broadcast and prune and the new technique, the delay is the minimum. Depending on the core selection, the distribution tree in CBT may not be the shortest path tree, which may result in larger delay. Increasing the network size, as shown in Fig. 23, increases the delay (since destinations are far from sources for larger networks). However, increasing the group size does not affect the delay as Fig. 24 demonstrates.

Fig. 25 and Fig. 26 give the bandwidth cost. In the simulation study, we assume that all links have the same cost. Calculating the cost then reduces to calculating the number of visited links when delivering a multicast traffic from the source to the destination. As Fig. 25 and Fig. 26 show, the costs in broadcast and prune and our new technique are identical and they are less than that of the CBT. Since broadcast and prune and the new technique build the same delivery tree, the cost should be the same for both approaches. The core selection, however, plays a major role in the resultant cost. As it is shown in the figures, a random selection of the core may result in larger cost. As expected, increasing either the network size or the group size increases the bandwidth cost since the data packets traverse more links.

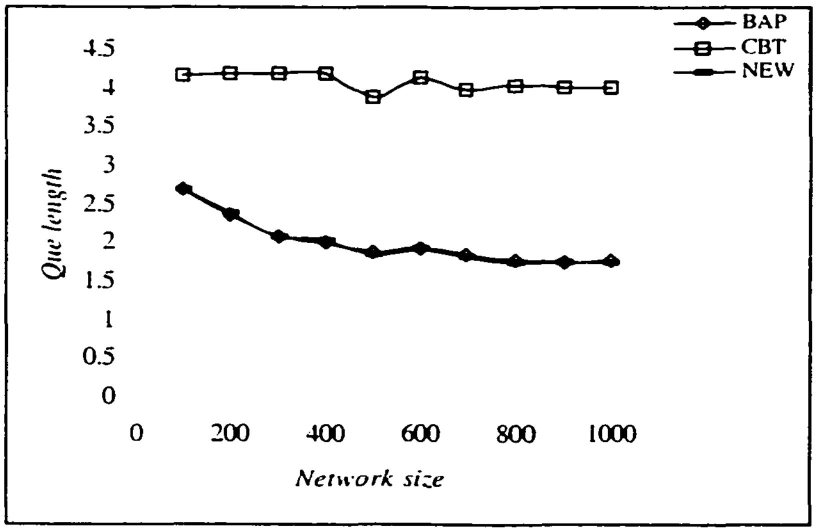

Fig. 27. Traffic concentration vs. network size.

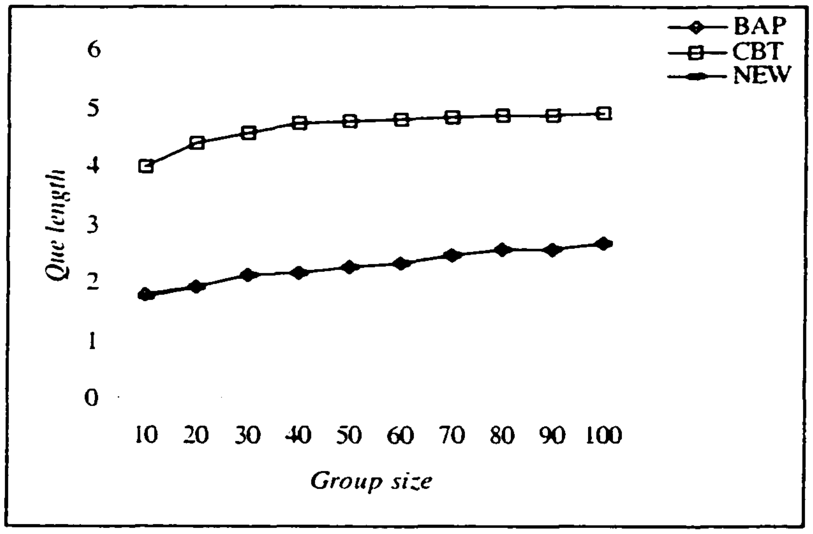

Fig. 28. Traffic concentration vs. group size.

To measure traffic concentration, we measure the average queue length for all routers. Fig. 27 and Fig. 28 give the traffic concentration for the three techniques. Since broadcast and prune and our new technique produce the same delivery tree for the same source and destinations, the traffic concentration is identical in both techniques. The traffic concentration in broadcast and prune and in our technique is much less than that of the CBT. In CBT, all traffic from all sources to destinations must go through the core, which significantly increases the traffic concentration around the core. Increasing the network size (Fig. 27) slightly decreases the concentration since the delivery tree is more flat with large networks, whereas increasing the group size (Fig. 28) increases the concentration as the tree is more condensed with large groups.

Fig. 29 and Fig. 30 demonstrate the average bandwidth overhead in building and maintaining the distribution trees for the three techniques. As shown in Fig. 29 and Fig. 30, the overhead in our technique is very close to that of the CBT and both are negligible when compared to the overhead in broadcast and prune. The reason of the large overhead in broadcast and prune is periodic broadcast of the multicast traffic across the entire network to build and maintain the distribution tree. When increasing the network size, and having a few and sparse set of destination, the overhead becomes very large. Increasing the group size (Fig. 30) does not have any effect on broadcast and prune, whereas it results in an unnoticeable increase in the other two techniques.

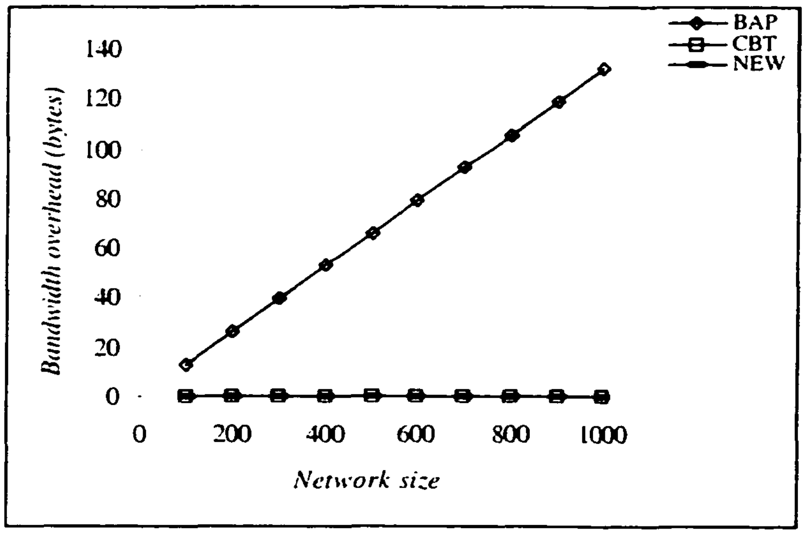

Fig. 29. Average bandwidth overhead vs. network size.

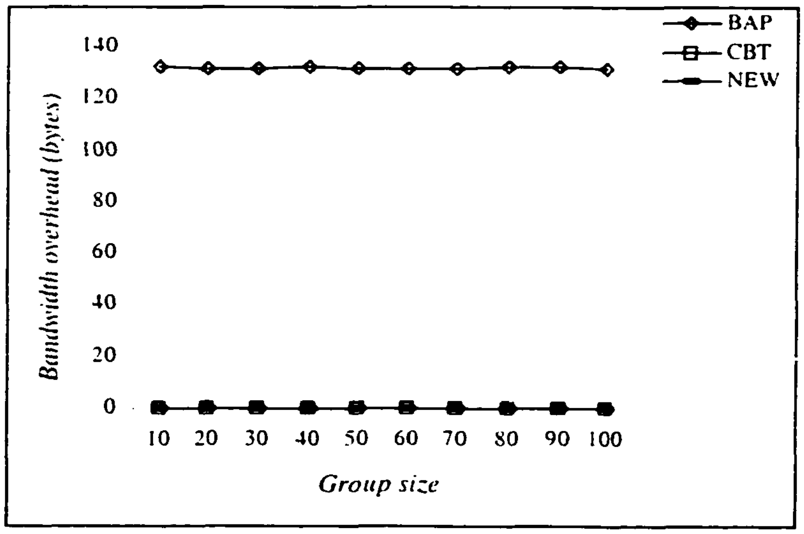

Fig. 30. Average bandwidth overhead vs. group size.

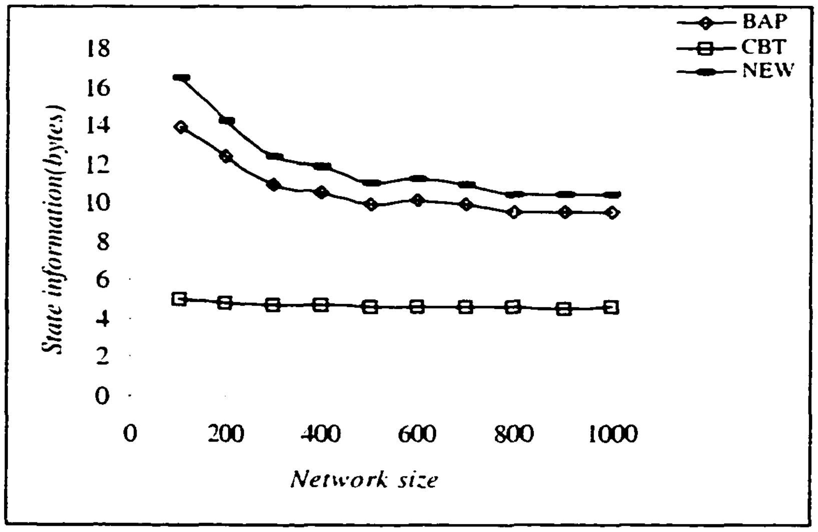

Fig. 31. Average memory usage vs. network size

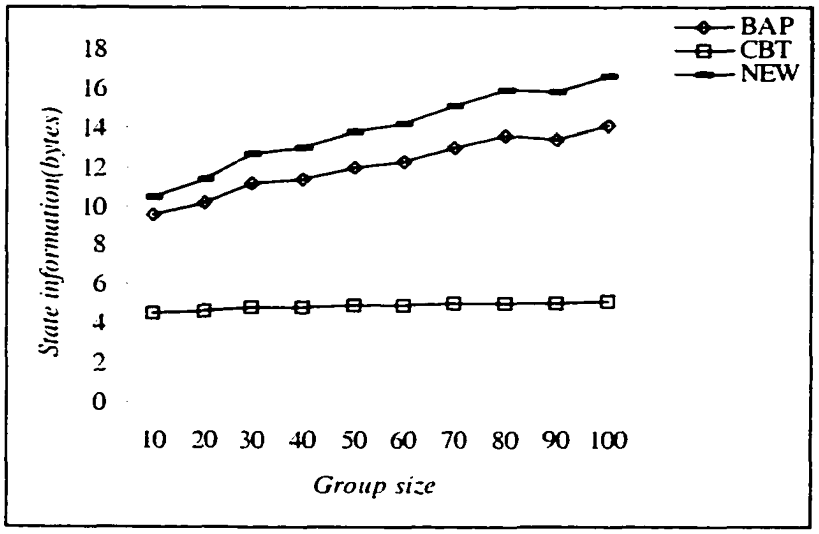

Fig. 32. Average memory usage vs. group size.

Fig. 31 and Fig. 32 show the memory usage within the network routers to store state information for the delivery tree. In broadcast and prune and in our new technique per group/source information should be saved in the router, whereas in CBT only per group information is stored. Moreover, in our new technique, information about the sources should be maintained. As shown in Fig. 31 and Fig. 32, memory usage in CBT is less than that of broadcast and prune. Moreover, memory usage of broadcast and prune is less than that of our new technique. Also Fig. 31 shows slight decrease in the memory usage in the three techniques. The reason behind this is for a small network size, the delivery tree is more condensed; a router in the route from the source to the destinations has more children than that of a larger network size (where the tree is less condensed). This increases the memory requirements for smaller networks. As expected, increasing the group (Fig. 32) increases the memory usage.

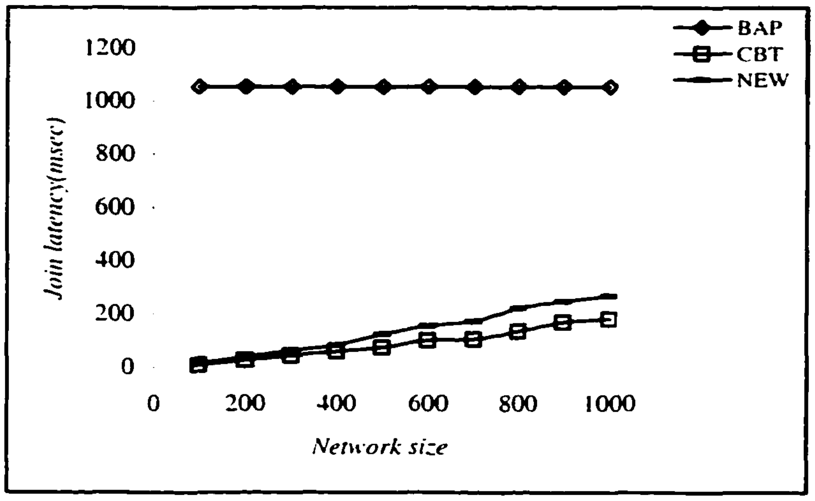

Fig. 33. Average join latency vs. network size.

The last performance index we consider in this section is the average join latency. As Fig. 33 and Fig. 34 show, the join latency of the CBT is the minimum of the three techniques. This result is due the fact that whenever a destination wants to join the multicast group, it sends a join message towards the core and the distribution tree is updated while the message is forwarded. In our new technique, in addition to the latency of forwarding the message from the destination to the core, there is also a delay in sending the information about the new members to other sources and in updating the distribution tree from the sources to the destinations. As for broadcast and prune, sources periodically broadcast the multicast traffic along the network edges to update the tree. On the average, the join latency is half the period that the sources take to update the tree. Broadcasting the traffic more often decreases the join latency while it increases the bandwidth overhead significantly.

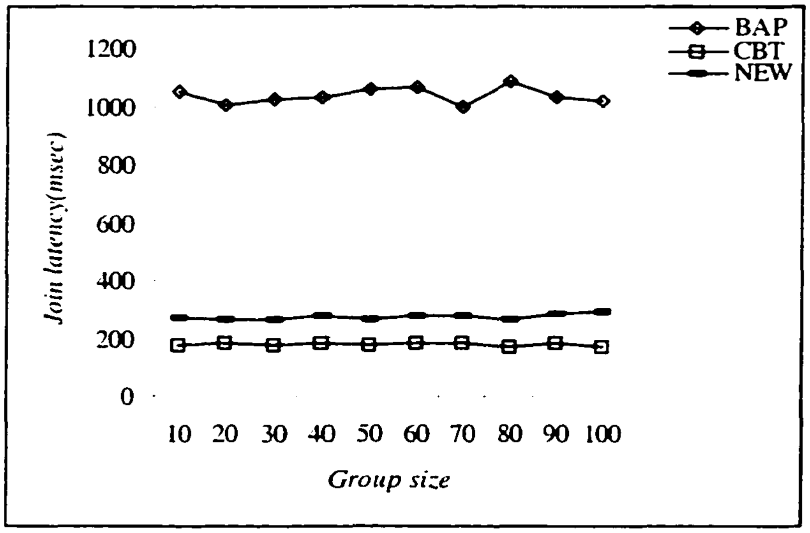

Fig. 34. Average join latency vs. group size.

From the above discussion we see that our new technique combines the advantages of both broadcast and prune and CBT. In particular, the new technique gives a minimum delay, minimum cost, and minimum traffic concentration similar to broadcast and prune. Moreover, it gives minimum bandwidth overhead similar to CBT. Memory usage and join latency, however, are worse than CBT. The rapid advances in computing technology lessen the memory usage factor, especially that the increase in memory usage is not significant. Also, when compared with CBT, the increase in join latency is insignificant, and is still better than the latency in broadcast and prune. TABLE 7 summarizes the performance of the three routing techniques.

TABLE 7

PERFORMANCE OF THE ROUTING TECHNIQUES

Performance measure	Broadcast and prune	Core based Tree	New approach
Average delay	Minimum	High	Minimum
Bandwidth cost	Acceptable	Depends on core selection	Acceptable
Traffic concentration	Low	High	Low
Bandwidth overhead	Very high	Negligible	Negligible
Memory usage	Per group/source	Per group	Per group/source
Join latency	Depends on the tree update period	Equals to message propagation delay from destination to core	Equals to message propagation delay from destination to core + from core to source

4.9 Conclusion

In this chapter we have investigated the problem of multicast routing for multimedia collaborative applications. We assume that the number of group members is not large and the membership dynamics is not fast, which are typical in most collaboration sessions. We have presented a new multicast routing technique that combines the advantages of broadcast and tree and core based tree. Specifically, our technique produces the minimum delay tree while the overhead in building and maintaining the multicast tree is minimal. We have studied the different aspects in our technique and we have modeled it using flow charts for the various entities involved. We also have introduced three variations of our technique to cope with the different cases of multicast collaborative applications. Finally, to evaluate our work we have presented a simulation study that contrasts the new technique with broadcast and prune and core based routing. Results of the simulation demonstrate the superiority of the new technique over the other two approaches.

CHAPTER V

MULTICAST CONGESTION AND FLOW CONTROL

This chapter investigates the problem of congestion and flow control for multicast traffic over datagram, packet switched networks and present an end-to-end solution to it. The focus of our study is on multimedia collaborative applications. Normally, the participants of a collaborative session span a heterogeneous inter-network. Moreover, the end systems may vary widely in their capabilities. Recently, two approaches have been introduced for multicast congestion and flow control: hierarchical multicast, which is window based, and multiple groups, which is rate based. Each approach has its advantages and disadvantages. In this chapter, we more explore the problem of multicast flow and congestion control and we introduce a new end-to-end technique that utilizes multiple groups and is window based. Our technique assumes a heterogeneous environment and minimizes the source overhead. To evaluate our work, we have conducted an analytical study that compares our technique with previous ones.

5.1 Introduction

The success of many components of multimedia collaborative applications depends heavily on the performance of the underlying network. For example, a timely delivery of the data to the different participants of a collaborative session is an essential to many components such as audio and video tools. Also, a low packet loss helps increase the quality of the perceived audio and video streams. In many situations some parts of the network may get congested, degrading the overall performance of the network. For best effort networks where there is no admission control for network traffic, all applications running over the network should cooperate to avoid congestion and control it would it happen.

In many situations, there is a variation in the capabilities of the end systems within a collaborative session. Assuming that the network provides an infinite bandwidth, coordination between the different sources and destinations within the session is still required so that sources avoid overwhelming the destinations with data. Sending with a rate higher than that the destination can handle results in a queue buildup at the destination and an eventual packet loss. The problem of regulating the traffic between the sources and destinations to avoid overwhelming the destinations with data is known as flow control.

The problems of congestion control and flow control are similar in the sense that both try to regulate the flow from the source. They, however, differ in their goal. On the one hand, congestion control tries to regulate the flow originated from the source in response to the network capabilities and its workload. On the other hand, flow control is concerned with the mismatch between the source and the destination and tries to regulate the flow of the source to meet the capabilities of the destination. Some networks can play a role in controlling congestion—examples include sending feedback from the routers to sources or dropping more packets from aggressive sources. The problem of flow control, however, is solely an end-to-end problem, since it depends on the sending and the receiving ends only, with a limited, if any, help from the network.

In this chapter, we consider end-to-end solutions for congestion and flow control in multicast communications for multimedia collaborative applications over datagram packet switched networks. Normally, the participants of a collaborative session span the entire network, which we assume heterogeneous (routers differ in capabilities and links differ in bandwidths) and dynamics (workload varies with time). The end systems comprising a collaboration session may also vary in their capabilities. We introduce a solution for controlling both the congestion and the flow by organizing the destinations into multiple multicast groups based on the capabilities of the destinations and their network paths from the source. Finally, we present an analytic study to evaluate our solution.

The rest of this chapter is organized as follows. General principles and some of the techniques of congestion control are given in Section 5.2. Section 5.3 is an introduction to multicast congestion control. Similar introductions for the general principles and the multicast issues in flow control are given in Section 5.4 and Section 5.5. In Section 5.6, we present our technique for multicast congestion and flow control. In Section 5.7, we give models for the congestion and the flow control problems and we present an analytical study that evaluates the various approaches to these problems. Section 5.8 presents the conclusions for this chapter.

5.2 Principles and Techniques of Congestion Control

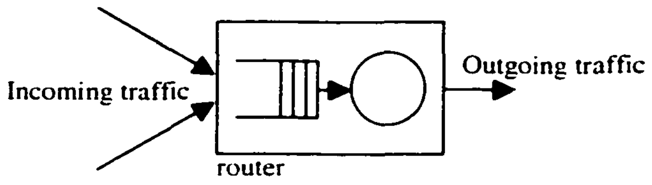

Fig. 35. Congestion: incoming traffic ≥ outgoing traffic.

We say that a network is congested when the demands exceed the available resources. In this situation, increasing the load does not increase the throughput of the network; rather, it dramatically decreases it [40], [29]. As Fig. 35 illustrates, congestion occurs when the incoming traffic at a node approaches or exceeds that of the outgoing traffic. In such a case, the length of the queue at this node grows indefinitely. Since routers' queues are of finite lengths, some of the incoming traffic may get dropped. The problem gets even worse when the sources try to compensate for the lost packets and level up their transmission rates. Increasing the queue size may not be of much help, as it was shown in a previous study that even with an infinite queue size congestion gets worse, since by the time packets arrive to the queue front they already have been timed out and duplicates have been sent [58].

Many solutions have been introduced to control network congestion [5], [15], [41], [55], [81]. Basically, congestion control schemes can be classified into open loop and closed loop [85]. Solutions based on the open loop approach do not monitor the dynamics of the network and they do not depend on any feedback from the congested spots. Instead, they try to prevent the problem of congestion from ever occurring. In order to prevent congestion, open loop protocols generally have mechanisms for admitting traffic into the network and mechanisms for packet dropping [26].

Closed loop congestion control solutions depend on monitoring the network and detecting congestion then passing a feedback message that gives congestion information to the sending end. The sending end uses the feedback message to adapt its transmission rate. Detection of congestion can be based on monitoring the network for the number of dropped packets, the average queue size, and the average packet delay. Congestion feedback can be explicit or implicit. Explicit feedback sends congestion information from the congestion spot to the sending end, while in implicit feedback the sending end conceives the presence of congestion along the path to a destination when an expected acknowledgement message is timed out or is received late [61].

In response to congestion information, sources have two approaches to control the outgoing traffic. The first is window based while the other is rate based. A window based approach try to limit the amount of data in transient. In this approach, each packet is assigned a sequence number. The source maintains a set of sequence numbers corresponding to the packets it is allowed to send. The source updates its set of sequence numbers when receiving acknowledges from the destination. The number of these packets can increase or decrease depending on the perceived status of the network. A window based technique can be considered as a closed loop controller that uses implicit feedback. This approach is suitable for best effort packet switched networks such as the Internet, where network routers do not provide much help in congestion control. An example of this approach is the sliding window mechanism used in TCP/IP [37].

Rate based approach, on the other hand, can be considered as open loop. Techniques following this approach try to regulate the average rate of data transmission by smoothing down the burstiness in the data. Generally, rate based approach works well in networks that support admission control (ATM network is an instance). Before start sending, the source and the network agree on a transmission rate, which the source sends at. An instance of rate based approach is the leaky bucket algorithm [80].

In controlling congestion, using a rate based or window based technique, a source tries to lower its throughput to meet that of the congested path. It should be noted that there is a trade off between time and quality. When choosing to sacrifice time, the source sends data with a slower rate, which results in longer delay. On the other hand, on sacrificing quality the source sends some of the data and drops some others. It is the semantic of the application that mandates what to choose. For example, video and audio tools may opt to choose delay over quality to meet their real time nature, while a shared white board may sacrifice the delay in favor of the quality.

5.3 Multicast Congestion Control

As introduced above, in unicast communication, a solution to the congestion problem is to feedback the source with information about the received packets at the receiving end. Information about lost packets and packet delays can be used as indications of congestion along the path from the source to the destination. The source can then adapt to network congestion based on this feedback. However, the problem of congestion control is magnified in multicast communication since traffic originates from a single end system and is distributed along many paths to many destinations. In a heterogeneous network, some of these paths may by congested while others may not, leaving the source with a problem to decide at which rate it should send [11]. Moreover, sending a feedback from all destinations to the source may result in a feedback implosion at the source. Fig. 36 contrasts congestion in unicast and multicast communications.

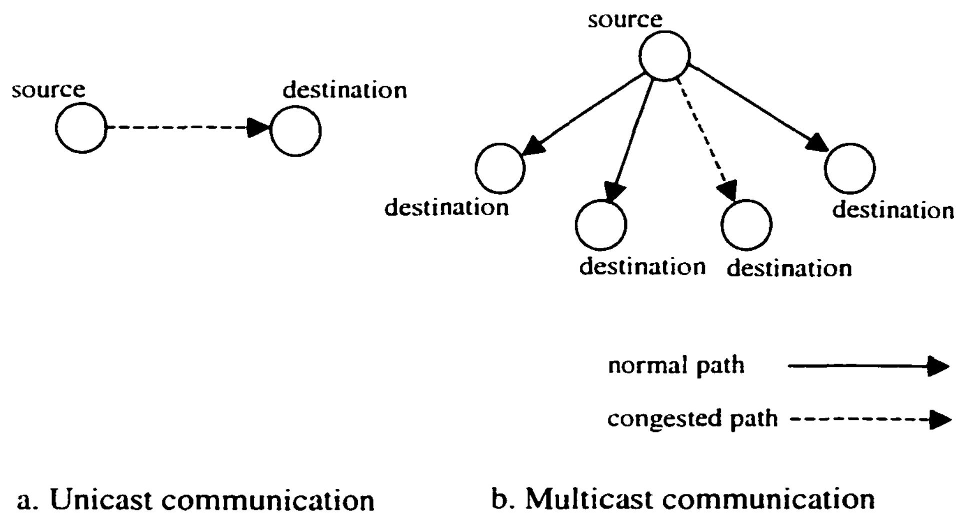

Fig. 36. Congestion in unicast and multicast communications.

In Fig. 36-b, only one links is congested while the others are normal. Three solutions exist to this problem: remove the destinations along the congested path from the collaboration session, adapt the sending rate to that of the slowest of the destinations, and send to each destination by its own rate. The first solution, removing the destinations along the congested path, may seem valid to solve some denial of service attacks. The solution, however, may not be acceptable in many applications where all participants of the collaboration session must remain. Because of its lack of applicability in many cases, this solution will not be considered any further in our work. The second solution (going with the slowest destination) is not fair as it may slow down destinations that do not suffer from any congestion. Moreover, and as a previous study has shown [30], this solution requires maintaining a window per each destination, as having only one window for all destinations unnecessarily restricts the throughput more than that is required by the congested path. While seems attractive, the third solution places a large overhead on the source in order to keep track of each destination and send to it in its rate. However, a modification to this approach by grouping destinations based on the condition of the network connections from the source to the destinations dramatically reduces such overhead. This solution will be investigated later in this chapter.

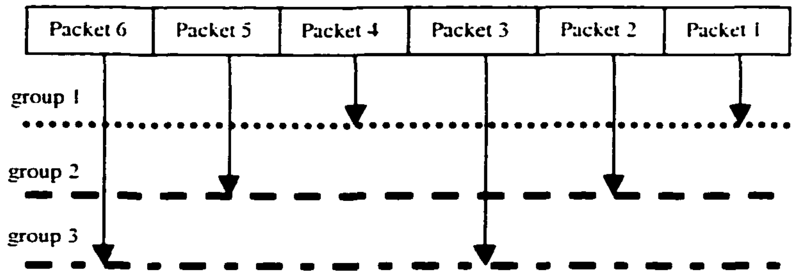

Fig. 37. Multiple groups for layered data.

Many techniques have been introduced to control congestion in multicast communications [30], [34], [70], [82]. Recently, two techniques have been introduced for end-to-end multicast congestion control. The first utilizes multiple multicast groups and the second is based on a hierarchical approach. The first is an open loop rate based technique that allows destinations to control the congestion [82]. The technique assumes that the data can be organized into layers. Each data layer then is oriented to a separate multicast group. As Fig. 37 shows, destinations join the appropriate number of layers that meet the available bandwidth and the transmission quality required. When detecting congestion in the network, a destination quits some of its layers. In order to help deciding which group a destination should join, the source multicasts probe messages periodically to destinations. This technique addresses the heterogeneity of the network and provides an efficient way to control congestion. The number of multicast groups, however, is limited since data can be organized into a few layers. Also, the technique addresses the problem of best effort applications where congestion is dealt with by sacrificing the quality of the data and does not provide an answer for reliable communications where packet retransmission is required. Moreover, it assumes that the application data can be organized in layers, an assumption that may not be valid for many applications.

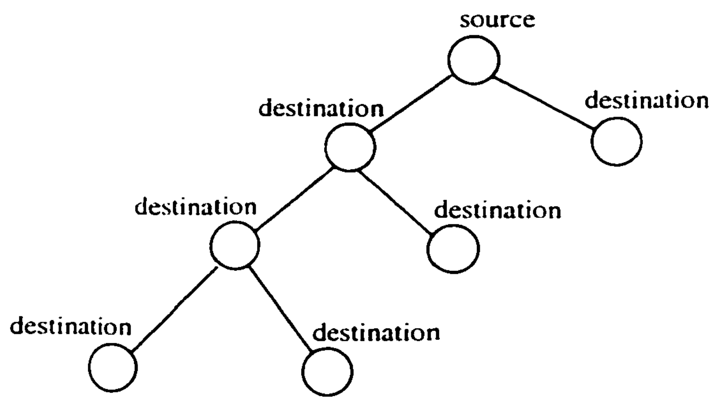

Fig. 38. Hierarchical organization for destinations.

The second approach is window based that maintains a window per each destination [30]. It organizes destinations in a tree-like structure, with the source at the root of the tree. Each parent keeps a separate window for each of its children and advances the window when it receives an acknowledgment from the corresponding child. The aggregate feedback then is directed by the parent to its immediate parent, and the process continues upward until it reaches the source. The main motivation behind this approach is to avoid feedback implosion at the source by letting some of the destinations (intermediate nodes within the tree) to handle some of the feedback. The technique, however, suffers from a large overhead in building and maintaining the tree. Moreover, it restricts the multicast session to proceed with the most congested path.

5.4 Principles of Flow Control

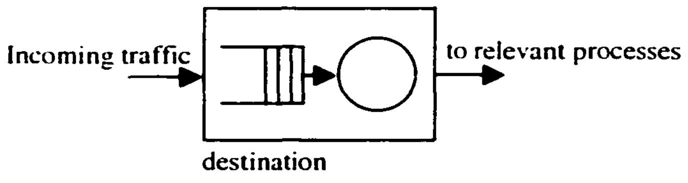

Fig. 39. Data buffering and processing at destinations.

Flow control concerns regulating the data sent from a fast source in order not to overrun a slow destination [47]. Typically, a destination allocates buffers to receive incoming packets. On receiving a packet, the transport layer of the destination must perform some processing before forwarding the packet to the appropriate process as shown in Fig. 39. If the packets mean arrival rate approaches the mean service rate, the system queue builds up indefinitely and the queue waiting time grows to infinity. Since there is a maximum length of the allocated buffers, data packets may get dropped off. The solution to this problem is to decrease the arrival rate of the data packets.

The problem of flow control is similar to that of end-to-end congestion control since both try to regulate the data flow sent by the source. They differ, however, in their goals—congestion control tries to avoid overwhelming the network with data whereas flow control tries to smooth down the mismatch between the source and the destinations capabilities. Even in the presence of a reliable network with infinite bandwidth, flow control is required to make sure that a fast sender (running on a fast machine) does not swamp a slow receiver (running on a relatively slow machine) with data packets.

Many techniques have been introduced for flow control. Since flow control is an end-to-end problem, all techniques that have been introduced to capture this problem rely on a mechanism to feedback the source with flow information existing within the destination. A simple, but inefficient, technique for flow is the stop-and-wait. In this technique the source sends its packet and waits for an acknowledgement from the destination. The source is not allowed to send any more packets until it receives the acknowledgement. The inefficiency of the technique lies in the waiting period, which makes poor utilization of the network and destination resources. Another approach to control the flow is the sliding window approach, which has been discussed in Section 5.2.

5.5 Multicast Flow Control

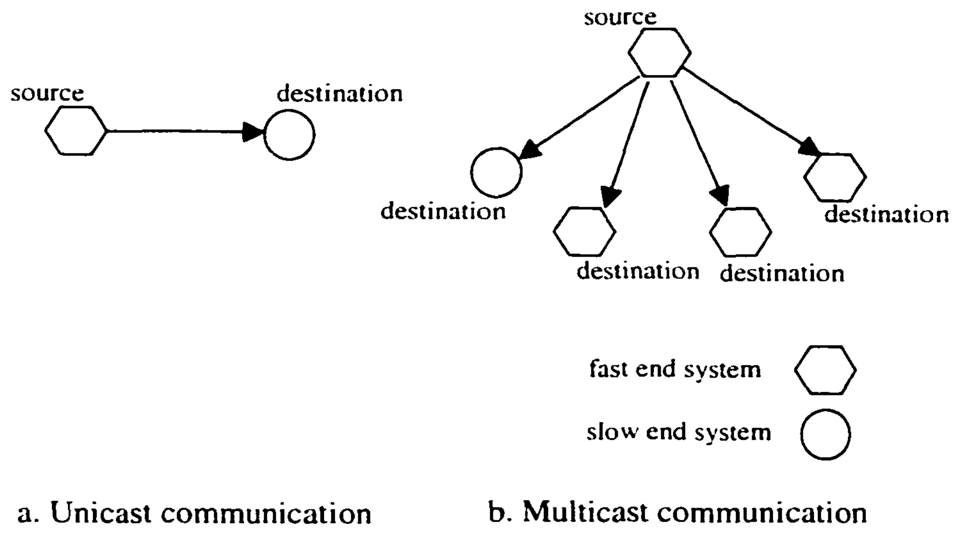

Fig. 40. End system heterogeneity in unicast and multicast communications.

Similar to congestion control, the problem of flow control in multicast is much tougher than that of unicast. Since there are many destinations with different capabilities, the source needs to decide which rate it should follow. Sending at the slowest destination rate may not be fair for faster ones. Fig. 40 demonstrates the effect of heterogeneity in multicast communications and contrasted to its unicast counterpart. Also, forwarding feedback from all destinations to the source may cause a feedback implosion at the source.

A technique that has been introduced for multicast flow control is using multiple multicast groups [10]. The technique is a rate based and assumes that the amount of data to be transferred is fixed, which is typical in many ftp applications. The goal of the technique is to multicast the fixed data from the source to the destinations in the minimum time possible. Destinations are organized into multicast groups based on their capabilities. Data are sent to each group independent of the other groups with a rate matching the destinations' capabilities. In order to work, the technique assumes a loss free network and data can be delivered out of order. Moreover, it assumes that the source knows the maximum rate it can send at for each destination. These assumptions, while being practical for some applications, do not fit well with multimedia collaborative applications running over best effort network. First, the network is not loss free. Second, assuming that the data can be sent out of order places a large buffering overhead on the destinations. Last, destinations overhead is dynamics and applications may produce data in bursts: sending at fixed rate may not be the best solution for such environment.

5.6 Multicast Congestion and Flow Control: A New Technique

In this section we give an end-to-end solution for the problems of congestion and flow control in multicast communication to support multimedia collaborative applications. Our solution addresses the heterogeneity problems in both the network and the destinations' computing and buffering capabilities. Our solution utilizes multiple multicasts to organize destinations into groups based on their connections to the source and their hosts' capabilities. The technique we introduce is window based in which the source maintains a window per group. To avoid feedback implosion, we adopt a hierarchical approach in which a representative is assigned for each group. A group representative is responsible for collecting feedback from the rest of its group members and sending the collective feedback to the source. To adapt to network and destination load dynamics, our solution allows destinations to migrate from one group to another and it permits groups' splitting and merging. Last, we consider the cases of real time and reliable traffics. There are three phases that can be considered when developing our technique: startup, steady state, and adapting to congestion. Each of these three phases is investigated in the following sections.

5.6.1 Startup Phase

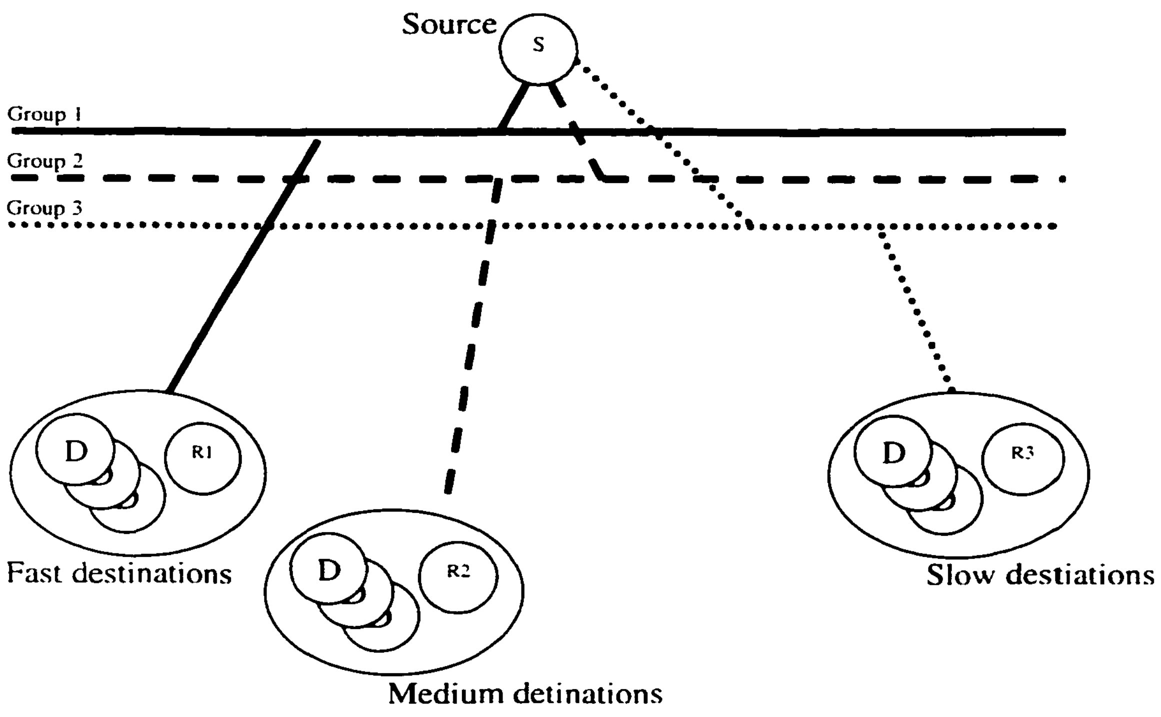

Fig. 41. Grouping destinations according to their capabilities.

At the startup phase, the source multicasts the first packet to all destinations and starts a timeout timer. The source, then, waits for feedback from all destinations. For each feedback it receives, the source calculates the round trip time for the destination the feedback received from. After the timeout timer expires or after receiving feedback from all destinations, the source categorizes the destinations to groups depending on their roundtrip times and assigns the fastest destination within each group as a representative for the group. For each group, the source announces the representative to the rest of the destinations in the group. Fig. 41 demonstrates how destinations can be grouped according to their capabilities and their connections to the source. As Fig. 41 shows, there is a representative (R) per group that acts as a point of communication with the source.

5.6.2 Steady State Phase

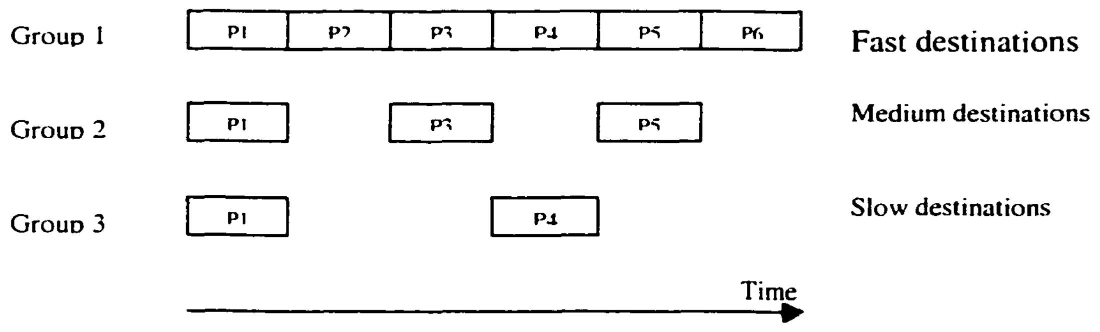

Fig. 42. Data distribution over groups for real time transmission.

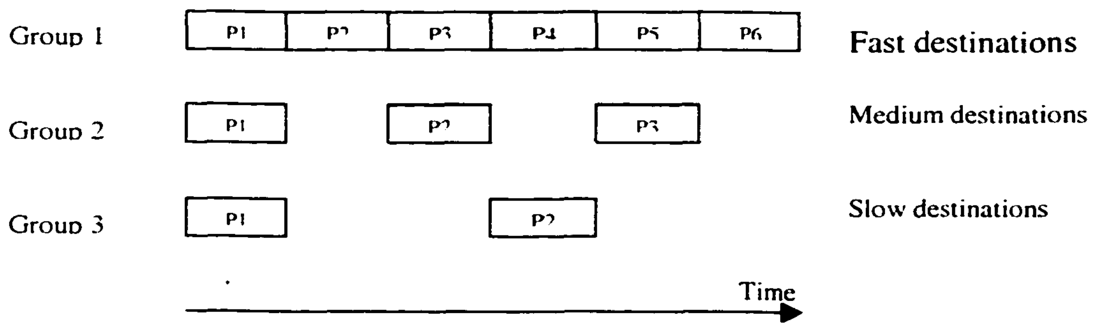

Fig. 43. Data distribution over groups for reliable transmission.

At the steady state phase, the source maintains a window per group. The window implementation depends mainly on the error control technique used and whether out of order delivery is allowed or not. The source sends each group its data independent of other groups. All destinations of a group send their feedback to the group representative, which in turn sends the collective feedback to the source. The source uses the representative feedback to adjust the group window. Periodically, the source sends the group representatives reports about the status of other groups (their window sizes and RTT) in order to provide representatives with information necessary to allow destination migration between groups, group merging, and group splitting. After organizing destinations into groups, the source multicasts each group its data based on the window information maintained for the group.

Our scheme considers two types of data transmissions: real time and reliable transmissions. Fig. 42 gives an example of packet transmission for real time applications such as video and audio, while Fig. 43 gives the same example for reliable transmission for the groups given in Fig. 41. In real time transmission, packets are dropped off for slow group (the penalty is the quality), while they are transmitted later in the reliable transmission (the penalty is delay). It should be noted that duplicate of the data are sent to the groups. In the worst case, each group contains only one destination and the situation degenerate to multiple unicast connections much similar to multiple TCP connections. In the best case, all destinations exist in one group, which can happen in homogeneous environments. Normally, the resultant situation lies between these two extremes.

5.6.3 Adapting to Variations in the Network and the Destinations Phase

Variation on network status and/or destinations' capabilities can be inferred from the round trip time and packet loss (which can be assumed by a missing feedback). Adapting to these variations can be done in two ways: intra-group adaptation and inter-group adaptation. Intra-group adaptation is required when most of the group members experience the same variation and can be performed by adjusting the group window maintained at the source. On the other hand, Inter-groups adaptation is required when few members of the group experience a variation that does not affect the rest. In this case, those members can move to another group, or can start another group (group splitting). Based on the information provided by the destinations and the source to the representatives of the groups, three cases may happen. First, a slow/fast destination can migrate to a slower/faster group if such group exists. Second, a group can be split to several groups. Third, two or more groups can be merged.

A destination can migrate to another group if the representative of the destination's current group determines that the destination does not belong to the group and should be moved to another one. This can happen if the destination workload or its network connection to the source has been changed. If a destination gets less capable, it should be migrated to a slower group in order to avoid slowing down the rest of its current group. After its migration to a slower group, the destination should ignore all packets it has already received in its previous faster group. The destination, however, should send its feedback to the new representative. On moving to a faster group, and in the case of reliable transmission, the destination should continue its membership in its previous slower group to receive the data that has been sent to the faster one. In the case of real time transmission, the destination joins the new group and quits the slow one.

Based on the information multicasted from the source to groups' representatives, groups can be split or merged. When many of members of the groups experience some variation that does not apply to the rest of the group, and provided that there no other group with the same capability of these destinations, group splitting is required. In this case, a new group is formed and a representative is selected and announced to the source. On the other hand, when the source detects two groups with the same capability—having the same window—the source asks the two groups to merge into one, and one of the two representatives is assigned as the new group representative.

5.7 Performance Evaluation

In order to evaluate the performance of our new technique we developed a simple, yet expressive and accurate analytical model of the congestion and flow control problems. The goal of any congestion and flow control technique is to avoid the network congestion and the mismatch between the source and the destinations while maximizing the throughput at the receiving end at the same time. Thus, we use the throughput at the receiving end as the performance measure in order to evaluate our technique and compare it against other techniques. In our evaluation study, we only consider the case of real time communications—retransmissions are not allowed. We calculate the throughput under three cases: no control, going with the slowest destination, and using multiple groups as given in our new technique.

In our model, we assume that there is a path from the source to each group. Every path is modeled as a server that has a queue of finite length. The service time, the arrival rate, and the maximum queue length differ from one server to another. Packets that are sent from the source towards a group of destinations go through the path from the source towards the group destinations, where the packets are queued and then served by the server within the path. This model can be used for congestion control as well as for flow control. For congestion control, the servers are the bottleneck routers along the path from the source to the destinations, whereas for flow control they are the buffering and processing modules in the transport layer within the destinations. An illustration of the model is given in Fig. 44. As Fig. 44 shows, there are *n* destinations that are organized into *S* groups. The source multicasts data to every group along its server (where the data packets are queued and served by the server along the path).

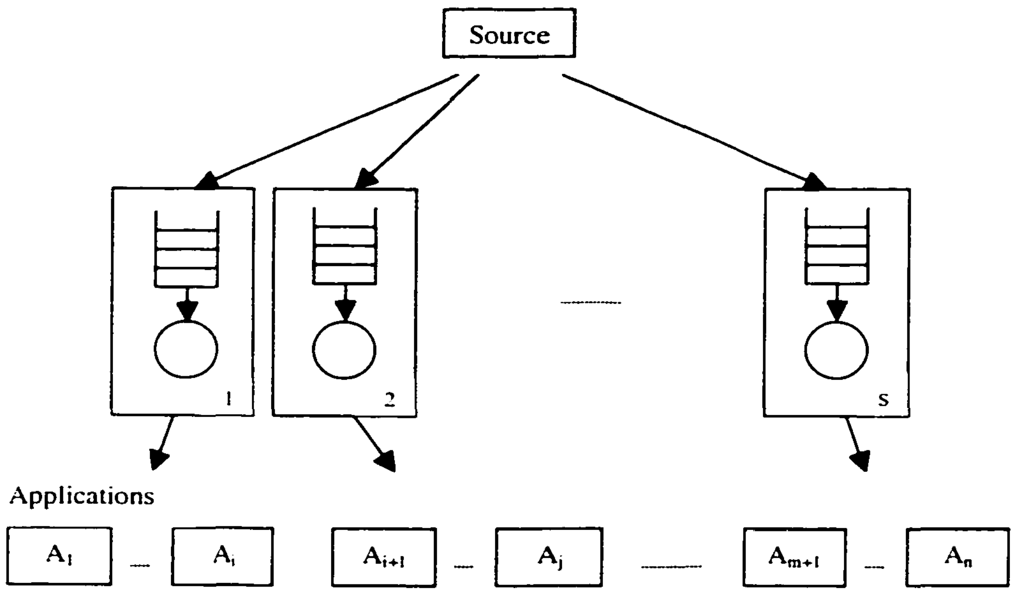

Fig. 44. System model.

The average throughput (λ_{avg}) is the total throughput for all destinations divided by the number of destinations. Given the destination throughput λ_i and the number of destinations n, the average throughput can be given as:

$$\lambda_{avg} = \frac{\sum_{i=1}^{n} \lambda_i}{n} \tag{8}$$

For unicast communication, increasing the sending rate increases the loss probability and consequently decreases the throughput. Assuming an M/M/1/N system—inter-arrival and service times are exponentially distributed, number of servers is 1, and buffer size is N packets—the loss probability can be related to the sending rate as follows [74]:

$$P = \frac{\rho'^{N}(1-\rho')}{1-\rho'^{N+1}} \tag{9}$$

Where

$$\rho' = \frac{\lambda}{\mu'} = \frac{\lambda/\mu}{1-P} \tag{10}$$

Equations (9) and (10) results in a nonlinear equation in *P*, which can be solved using iteration or trial and error.

In real time, it is most likely that retransmission of lost packets is not allowed since the delay encountered in the retransmission is not affordable. In the case of no control, the throughput is limited by the capacities of the links from the source to destinations. Hence, the average throughput per destination can be given as:

$$\lambda_{avg} = \frac{\sum_{i=1}^{n} MIN(\lambda_s, \lambda_i) * (1-P_i)}{n} \tag{11}$$

Where:

λ_s is the sending rate

λ_i is the capacity of the link from the source to destination i

n is the number of destinations

P_i is the loss probability of path i and is given in equation (10)

As equation (11) illustrates, increasing the sending rate increases the received throughput until the capacities of the links and/or the destinations' processing powers are reached. Beyond this limit, increasing the sending rate results only on wasting the network bandwidth since the extra packets will be dropped at the bottlenecks.

Going with the slowest destination, the throughput is limited to the slowest destination. Hence, the average throughput per destination is:

$$\lambda_{avg} = \lambda_s * (1 - P) \qquad (12)$$

Where:

λ_s is the sending rate and it is $\leq \lambda_{min}$ (the minimum link capacity from the source to all destinations)

P_t is the loss probability of the bottleneck path in the group

Equation (12) shows the performance of going with the slowest destination approach. In this case, the average throughput is limited with the most severe bottleneck.

Using our new approach, the average throughput per destination is:

$$\lambda_{avg} = \frac{\sum_{i=1}^{g} MIN\left(\lambda_s, m_i \times MIN_{j=1}^{m_i}(\lambda_j)\right) * (1 - P_i)}{n} \qquad (13)$$

Where:

λ_s is the sending rate

λ_j is the capacity of the link from the source to destination j

g is the number of groups

m_i is the number of destinations in group i

n is the number of destinations

P_i is the loss probability for the bottleneck path in group i

Equation (13) gives the throughput achieved by our new approach. The throughput per group is limited to the slowest destination in the group and the overall throughput is given as the average of groups' throughputs.

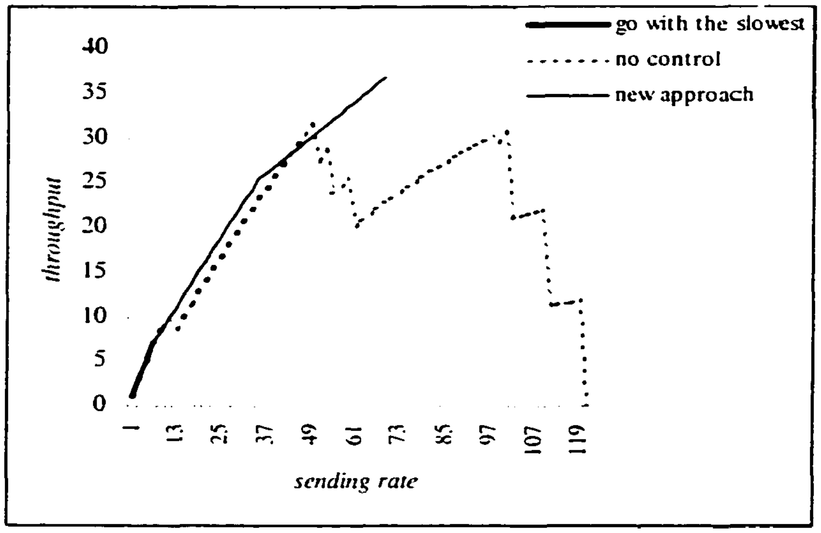

Fig. 45. Throughput in real time transmission.

Fig. 45 compares our approach against the no control and the going with the slowest approaches. The figure assumes 9 destinations with capacities of 10, 12, 13, 50, 55, 60, 100, 120, and 110 K bytes/second. All destinations have the same buffer size (8 packets). When grouping using the new approach, we assume three groups that organize destinations based on their capacities. Also, we assume that the source limits its sending rate to be less than the capacity of the corresponding links. As Fig. 45 shows, limiting the sending rate to the slowest limits the throughput to a very small number, wasting the capacities of the other destinations. Adopting a no control approach brings the throughput to zero for the destinations that have sending rate approaching the capacities of the links (the stair case effect shown in the figure is due to this phenomenon). Using our new approach, the throughput obtained is much better than that obtained when going with the slowest destinations or that achieved if we do no have any control.

5.8 Conclusion

In this chapter we have studied the problems of congestion and flow control in multicast communication and we have introduced a new technique based on using multiple multicast groups to solve both problems. Our new technique is window based that establishes a window per group. Data are sent to every group independent of the other groups. In order to avoid overloading the source with destinations' feedback, a representative per group is designated to collect the feedback from the rest of the group members and relay the feedback to the source. We have introduced solutions to allow destinations to migrate between groups. We also have presented techniques for merging and splitting groups. To evaluate our approach, we have presented an analytical study that compares the new technique with the cases of going with the slowest destination and the no control mechanism in real time communications where retransmissions are not allowed. Results of our study shows that our technique performs much better than the other two approaches.

CHAPTER VI

MULTICAST ERROR CONTROL

An error control scheme is an essential to the communication layer for multimedia collaborative application. As introduced earlier, multimedia collaborative applications consist of several components that may vary in the error constraints they place in the communication layer. For example, video and audio may perform reasonably well even in the presence of some errors. However, the quality for these two tools increases when the experienced errors decrease. On the other hand, some other components, shared whiteboard for example, require a reliable data transmission. The aim of this chapter is to study end-to-end multicast error control in a heterogeneous environment for multimedia collaborative applications. Mainly, end-to-end multicast error control techniques fall in one of two categories: automatic repeat request (ARQ) and forward error correction (FEC). ARQ relies on error detection and retransmission [27], [66], while FEC is based on sending redundant codes (along with the data) that can be used to recover from the errors experienced [35], [65]. Both approaches have their strengths and weaknesses. Recently, a hybrid technique has been introduced to combine the advantages and avoid the shortcomings of ARQ and FEC [59]. In a heterogeneous environment, not all end systems and their network connections to the source have the same capabilities. Consequently, destinations may experience different error patterns. In all error control techniques, error control transmissions are needed (whether retransmissions or repair codes). Since not all destinations encounter the same errors, directing the error control transmissions to all multicast group members wastes the resources of the unaffected destinations. A technique to overcome this problem is the use of multiple multicast groups to deliver the error control transmissions only to the interested destinations. While eliminating the unneeded processing of error control transmissions from the destinations, this technique places an overhead on the network in maintaining the delivery trees for the extra multicast groups. In this work, we investigate end-to-end multicast error control in heterogeneous environment using multiple multicast groups. In particular, we enhance an early work in utilizing multiple groups in ARQ and we introduce new techniques for using multiple groups in FEC and hybrid approaches. In our work, we consider the effect of multicast routing: an important factor that has been ignored by previous studies. We present a simulation study to evaluate the performance of our work.

6.1 Introduction

This chapter's goal is to study end-to-end multicast error control in heterogeneous environment for multimedia collaborative applications. When built atop a packet switched, datagram, best effort network such as the Internet [67], the multicast transport layer carries on the burden of handling the errors that may have resulted during data transmission. Examples of these errors are packet loss, duplicates, and out of order delivery. In end-to-end error control, the responsibility of repairing these errors is placed on the end systems—network routers are not assumed to play any role in the process.

For a large inter-network such as the Internet, end systems, network routers, and communication links widely vary in their capabilities and bandwidths. Such a heterogeneous environment raises a challenge in developing the required error control service for multicast communication. In this environment, destinations may encounter different error patterns. The problem is even magnified for large group sizes. Since end systems may encounter different error patterns, directing error control transmissions to the whole group members has the effect of wasting resources of the network as well as the destinations in delivering and processing unneeded packets.

Many techniques have been introduced to counter this problem. Basically there are 4 approaches to relief destinations and network from unneeded error control transmissions. The first uses local recovery to isolate areas of errors, normally at the domain level, and limit error control transmissions within these areas. While local recovery reduces the effect of unneeded transmissions, it still does not eliminate it within the isolated areas. A representative for this approach is SRM and its local recovery enhancement [27].

Another approach is to use hierarchical error control techniques. In this approach, destinations are hierarchically organized, and the nearest possible node to the error location carries out the error recovery procedure. As in local recovery, the approach reduces the effect of unneeded error control transmissions but does not eliminate it. Examples of reliable multicast systems based on this approach include log based reliable multicast [33] and reliable multicast transport protocol [63].

The third approach is based on active networks. The approach places some of the error control functionality over the network routers by letting them responsible of suppressing the unneeded error control transmissions. An example of this approach is active reliable multicast [50]. This approach is outside the scope of this thesis, which only considers end-to-end error control techniques.

The last approach is using multiple groups to carry the error control transmissions. A technique that has shown promising results in multicast ARQ is utilizing multiple groups for retransmissions [45]. The basic idea is to have retransmissions sent over multicast groups other than the group that carries the original data. Upon error detection, a destination joins the group that carries the required retransmission, and then it leaves the group after packet reception. The study examined having one group for every retransmission (which assumes an infinite number of groups) and concluded by examining the more practical case of sending several retransmissions over one group. Either way, the study showed a saving in the destinations' computational effort over techniques that use only one group for both the data and retransmissions.

While using multiple groups for retransmissions enhances the performance of the destinations as has been shown in [45], it places an overhead on the network in maintaining the extra groups. Another paid price is the increased latency, as the retransmission process cannot start until the multicast routing tree for the retransmission group has been built. These two deficiencies depend greatly on the routing protocol deployed in the network. Previous work has ignored the impact of having multiple groups on the network and the error repair latency.

From the above discussion, local recovery and hierarchical error control techniques reduce the effect of unneeded error control transmissions but does not eliminate it. Active reliable multicast involves network routers in the procedure, a task that seems difficult regarding the current status of the multicast communication technology. Utilizing multiple groups in multicast error control has the potential to provide a solution for the problem, and is the focus of our work.

The contribution of our work is twofold. First, we propose an enhancement for utilizing multiple groups in multicast ARQ. The basic idea is to let destinations continue their memberships in the retransmission groups, rather than leaving the group immediately as suggested in [45], for a period of time that depends on the error rate experienced. To do so, we restrict the number of retransmission groups to one. While this technique may waste some of the destinations' resources in processing unneeded retransmissions, it decreases the network overhead and the overall latency introduced by the error control procedure. We have conducted a simulation study to compare our approach with the previous work of [45]. The results of our study indicate a better network overhead and latency, while destinations' overhead is slightly increased when using our proposed technique.

Second, we propose using multiple groups in FEC and hybrid multicast error control. For FEC, we assign multiple groups for the error correction packets. Every group is capable of repairing a predefined error rate, i.e. groups are identified by the error rates they can repair. Destinations join these groups based on the error rate they encounter. For example, one group may be dedicated to repair a 10% error rate and another to rectify 20%. Destinations that encounter a 10% error rate may need to join the first group, whereas destinations encountering a 20% error rate may need to join the second group. For those encountering a 30% error rate, they may need to join both groups. We have conducted a simulation study that compares our approach with that using only one group for both data and codes. Our simulation results show a decreasing load at the destinations, while latency is slightly increased. The network bandwidth consumed in building and updating the routing trees for the multiple groups can be kept minimal by limiting the number of groups to a small number and slowing down the dynamic of membership over these groups.

For Hybrid techniques, we propose having multiple groups for error correction packets, and one group to carry retransmissions. As with FEC, destinations join the error correction code groups based on the error rate they encounter. If a retransmission is required, a destination joins the retransmission group and continues its membership for a period of time proportional to its error rate. We also have conducted a simulation comparison between our technique and the one that does not deploy multiple groups. The obtained results show an improvement in destinations' workload and network bandwidth for our technique over using one group for data, correction codes, and retransmissions.

The rest of this chapter is organized as follows. Section 6.2 is an introduction to error control in multicast communications. Section 6.3 is devoted for the performance metrics of multicast error control techniques. System model and simulation measurements are introduced in Section 6.4. Section 6.5 is dedicated for utilizing multiple groups in multicast ARQ. In this section, we present previous work in this area, then, we introduce an improvement for these techniques and present our simulation study. In Section 6.6, we introduce a new technique that utilizes multiple groups in FEC. In Section 6.7, we present a new technique using multiple groups in hybrid error control techniques. Section 6.8 gives the relationship between the error control module and the congestion and flow control module. Section 6.9 concludes this chapter.

6.2 Multicast Error Control

As introduced above, error control techniques fall in one of two categories: automatic repeat request (ARQ) and forward error correction (FEC). ARQ is based on error detection and recovery. A main advantage for such technique is that the amount of redundant data is a minimal as retransmissions are only triggered with the detection of errors. ARQ techniques, however, generally suffer from large delay that renders them unsuitable for many real time applications such as video and audio. For other applications that do not function with errors while delay is not at the same level as reliability, example includes shared whiteboard, ARQ is an attractive choice.

In FEC, redundant data codes are computed and sent over along with the original data. Destinations are responsible for error detection and recovery from the error using the already sent correction codes. The main advantage for FEC is the elimination of the large retransmission delay, which makes it attractive for audio and video applications. However, the cost of computing the error codes at both the sender and the destinations is very expensive. Also, since coding is performed and sent before any error occurrence, determining the required amount of correction codes relative to the original data is a challenge, especially in a heterogeneous environment.

Both ARQ and FEC have their strengths and shortcomings. Recently, a new hybrid approach has been emerged to capture the advantages and avoid the shortcomings of both ARQ and FEC techniques. The main idea is to have a fixed amount of coded data relative to the original data. Errors beyond the capabilities of FEC are recovered using retransmissions. This approach has been shown to avoid the problem of deciding the amount of correction codes and to lower the number of retransmissions. The rest of this section gives introduction to multicast ARQ, FEC, and hybrid error control techniques.

6.2.1 Multicast ARQ

In ARQ, retransmissions of the data are triggered upon the detection of data loss. Mainly, there are three questions that face this approach:

1. Who is responsible for error detection?
2. Who carries on the retransmissions?
3. To whom should the data be retransmitted?

The responsibility error detection can be placed on the sender, the destinations, or both the sender and the destinations [79], [84]. Error detection responsibility can be placed on the sender by, for example, requiring all destinations to ACK every multicast packet. In such mechanism, the sender safely can assume no error if it receives ACKs from all destinations; otherwise it concludes an error has occurred. This approach is known as sender initiated multicast. An example of this approach is [6]. Error detection can also be placed on the destinations. An example for such approach is to require the sender to stamp every packet with a sequence number. A destination can detect an error if there is a break in the sequence numbers it has received. This approach is known as receiver initiated multicast. An Instance of this technique is [48]. Finally, both approaches can be combined to place the error detection responsibilities on the sender and the destinations [66], [7].

The task of error recovery can be placed over the sender, the destinations, or special servers. Placing the responsibility of error recovery on the sender can be done, for example, by requiring the sender to start a retransmission procedure upon the detection of errors [6]. Another alternative is to let some of the destinations who received the data perform the retransmissions upon error detection [60]. Recently, a server oriented error recovery approach has been investigated to provide a better performance [44].

Upon error detection, retransmissions can be oriented to the whole group or only to the destinations that encountered the error. In the former case, one multicast group is sufficient to perform the delivery of both the original data and the retransmissions. In the later case, beside the original data groups, other groups are required for the retransmissions. Destinations may join and leave the retransmission groups dynamically according to the error pattern they encounter [45].

The answers of these three questions greatly affect the performance of the system. For instance, if the responsibility of error detection is placed on the sender solely, the sender may get overloaded since all destinations report their status through acknowledgments (ACKs), a situation known as ACK implosion. Also, if retransmissions are sent to the whole group, computational power of the destinations that have not encountered errors and the bandwidth along the paths from the sender to these destinations are unnecessarily wasted.

6.2.2 Multicast FEC

FEC is based on sending repair codes along with the data. Initially, FEC was applied at the bit level, i.e. bit errors can be recovered using FEC. As most of the errors now a day are at the packet level (in the form of lost packets), the trend is to use FEC at the packet level. Fig. 46 illustrates how FEC works. Initially, the sender computes *m* packet codes (2 in Fig. 46) for every *n* data packets (5 in Fig. 46). The sender then sends the codes along with the data (*n+m* packets). Upon receiving any *n* (data or codes) packets, the destination inputs the received packet into its decoder, which extracts the original *n* data packets [46], [59], [72].

The main advantage of FEC is the elimination of the delay of retransmissions, which makes it attractive for real time applications. However, FEC is faced with two questions. First, the number of coded packets has to be decided before any occurrence of errors. Sending too many packets wastes the computation resources in the coding and decoding processes at the sender and the destinations and wastes the network bandwidth in carrying unneeded code packets. Sending few coding packets may not achieve the level of error control desired. The second question that faces FEC is how to code and decode the correction packets. The cost of the coding and decoding processes is high. Many techniques have been introduced for packet level coding and decoding. One technique is to exclusive or (XOR) the data packets to produce the correction codes. The main problem of the coding and decoding processes is that they are computationally expensive. For more discussions on the coding and decoding of the repair codes the reader is referred to [13], [71].

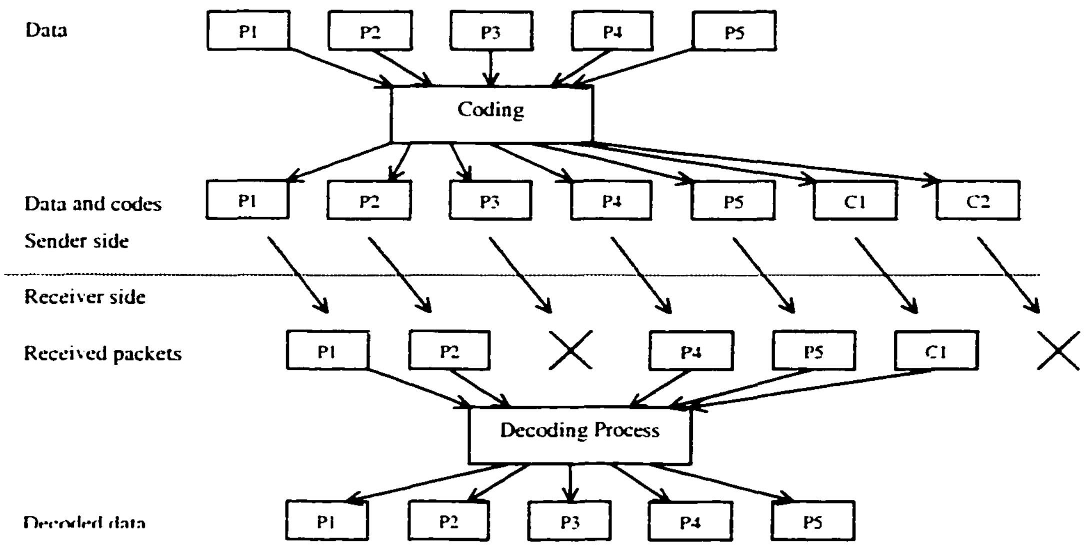

Fig. 46. Example of FEC error control.

6.2.3 Hybrid Techniques

To overcome the disadvantages that face both ARQ and FEC and to combine their advantages, hybrid techniques have been developed [73]. The basic idea of hybrid techniques is that correction code packets are sent along with the data packets. If the correction codes are not enough to reconstruct the data, retransmissions are used. Mainly there are two techniques to organize ARQ and FEC together: layered and integrated. In the layered case; shown in Fig. 47 (a); retransmissions are also sent using FEC, meaning that correction codes are computed for the retransmissions. On the other hand, the integrated technique retransmissions are sent without further coding, which is given in Fig. 47 (b). It is worth noting that in sending retransmissions, the exact lost packet is not required, rather and correction code packets may be sufficient to reconstruct the original data.

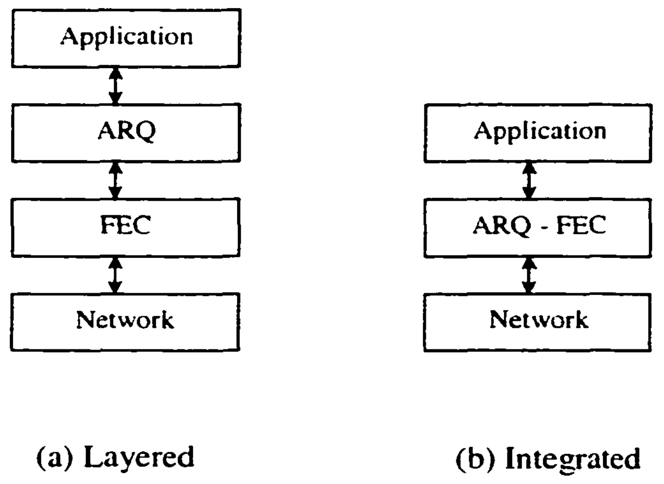

Fig. 47. Hybrid techniques.

Using hybrid techniques has the effect of decreasing the number of retransmissions and getting around the problem of deciding how many correction packets are required. Also, different lost packets at different destinations can be repaired using same correction codes. For example, assume a number of destinations, each has one packet lost (not the same packet), only one correction packet may be used to repair the errors at all destinations, an improvement over using ARQ only which requires sending as many as the different lost packets.

6.3 Performance Metrics

Error control techniques can be measured according to many criteria. The most important measures, however, are latency, scalability, and overhead placed at the sender, the destinations, and the network. Using an error control mechanism introduces latency due to several reasons. Processing ACK or NACK packets, processing retransmissions, and computing correction codes are some causes for such latency. Many applications require a real time—hard or soft—performance. In such cases, the latency introduced due to error control should be a minimal.

The second measure is scalability. A scalable error control mechanism is the one that does not place an upper limit on the group size due its functionality. An error control mechanism may put an upper limit of the group size. For example, using ACK for every packet from all destinations to the sender in ARQ may limit the group size to the number of ACKs the sender can process at the same time. Some applications may involve hundreds of thousands of users, a news broadcast is an example. In such systems, scalability is a main concern.

The third measure is the overhead placed on the end systems of the sender and the destinations and the network itself. For the end systems, this overhead comes in the form of state kept about other end systems and processing packets of ACKs, NACKs, retransmissions, or correction codes. Keeping the overhead minimal usually leads to a better delay and scalability. For the network, the overhead comes at the cost bandwidth consumed. Some techniques place some of the error control mechanisms on the network routers. Keeping network overhead small helps getting a better service for the multicast applications as well as for the whole network community.

6.4 System Model and Simulation Measurements

This work assumes a multicast enabled network, where multicast follows the host group model [18], [19]. In this model, destinations interested in receiving multicast traffic join a group identified by a unique address. Traffic addressed to a group is delivered to the group's destinations; senders need not know the destinations' identities nor do they need join the group to send to it. This model places no restriction on the group size and membership.

In conducting our simulation study, we use a Bernoulli distribution to decide whether a packet is received or lost at a destination. We assume that loss events at the destinations are mutually independent. Every packet has a loss probability, which is assumed to be a uniform random variate and is independent of any destination. All simulation programs are written in *C++* and use the multiplicative linear congruential method to generate pseudo random numbers from which the uniform random variates is generated. For a discussion in random variates, the reader is referred to [39].

The simulation studies conducted in this thesis do not assume any specific error control mechanism. Rather, we abstract our measurements in order to provide a general, yet concrete comparisons between different approaches. As introduced earlier, there are 4 performance metrics for multicast error control. Since the introduced latency by using multiple groups is primarily due to building and maintaining the routing tree for these groups, the measurement unit is chosen to be the time required for building the tree. The average data loss is calculated over all destinations.

Since using multiple groups does not place any load on the sender (senders need no join a multicast group to send to it), it does not increase, nor does it reduce the system scalability. System scalability can be enhanced by other techniques such as receiver-initiated multicast or local recovery. Based on this, scalability is not considered in our measurements. As for the overhead, same argument as above holds for sender's overhead. Destinations' overhead is measured by the number of unneeded packets received, which is an indicative and protocol independent way of measuring the overhead.

Measuring network bandwidth overhead, however, constitutes a challenge since we are trying to achieve two objectives that seem to be contradictive. The first objective is not to assume any specific topology in order to get a general measurement and the second is to get a correct and representative measurement, which seems not to be achieved unless we have a specific topology. In order to solve this problem, we treat the two types of routing techniques (broadcast and prune, and CBT) separately. We assume that the tree in broadcast and prune is dense (that is, the destinations' distribution is dense in the tree) and sparse CBT. In both broadcast and prune and CBT, an unneeded packet is assumed to waste a fraction of the network bandwidth equal to the number of destinations unnecessarily receiving this packet divided by the group size. On the one hand, building or maintaining the tree in broadcast and prune is assumed to waste what is equivalent to the total network bandwidth (since the packet is broadcasted along the network edges). On the other hand, building and maintaining the tree in CBT is assumed to waste a small bandwidth fraction (in sending join and leave messages along the tree). These assumptions provide an abstraction, which produces results that are general and indicative at the same time.

6.5 Multiple Multicast Groups in ARQ

Utilizing multiple groups in multicast error control relies on using one group for the original data (which we call the main group) and some others for the retransmissions (which we refer to as the auxiliary groups). Destinations join the auxiliary groups dynamically only when needed.

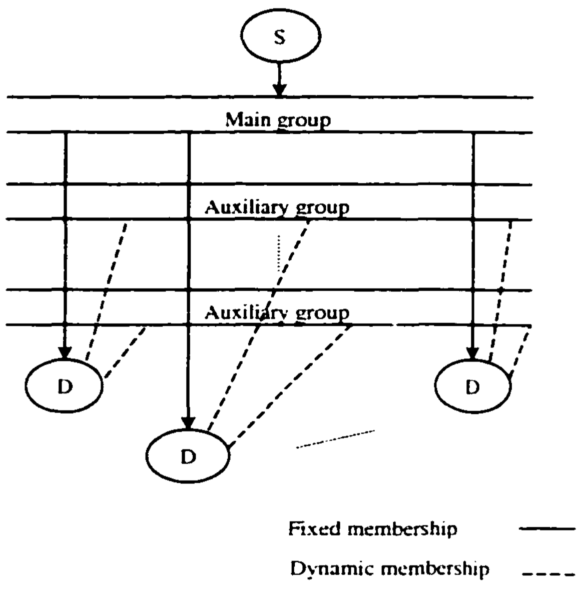

Fig. 48. ARQ error control using multiple groups.

Mainly, there are two issues that need to be resolved: how many auxiliary groups are required and how they are formed. To completely remove the destinations' overhead, one group is needed for every retransmission (i.e., having infinite number of groups). However, it has been shown that having a limited number of groups can give most of the benefit of having infinite number of groups [45]. The second issue is how groups can be formed. This issue by itself consists of two points. The first is how to identify these groups (determining the multicast groups' addresses). Second, who should join/leave these groups and when they should do so. In this dissertation, we identify two mechanism of group formation: packet grouping and error rate grouping.

Having an infinite number of groups, groups' addresses can be calculated as functions of the packet sequence numbers. However, when the number of groups is limited, groups' addresses can be defined beforehand and packets can be mapped to groups based on their sequence numbers. In either case, destinations requiring retransmissions join the appropriate groups and leave them immediately after receiving the required retransmissions. We call this technique packet grouping as destinations are grouped based on the packet's sequence numbers. Fig. 48 gives the architecture of packet grouping. As the figure shows, beside the main group, which carries the original data, there are a number of auxiliary groups that carry the retransmissions. Destinations join the main group all the time and they join and leave the auxiliary groups dynamically when needed.

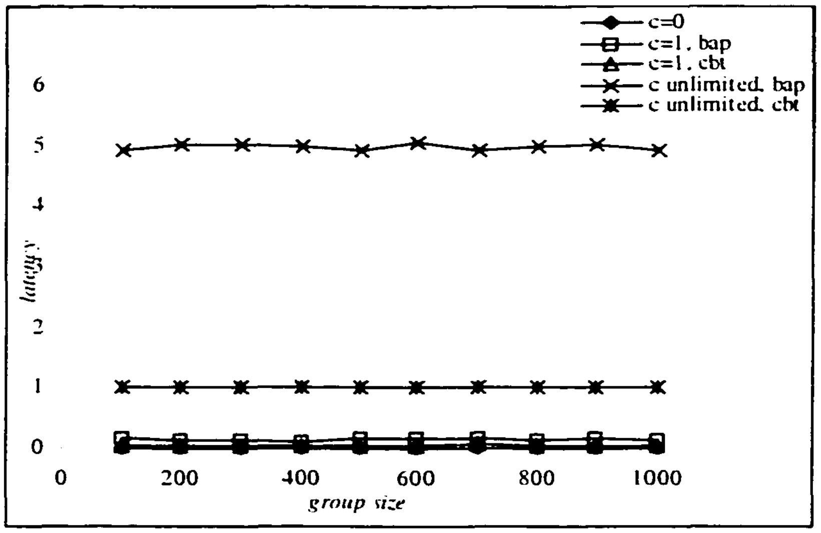

Fig. 49. Latency in ARQ.

A major difficulty that faces the packet grouping technique is that group membership dynamics is very fast. This places an overhead on the network routers and consumes a large bandwidth in building and maintaining the routing tree for the auxiliary groups, especially if the deployed routing technique is a broadcast and prune. To decrease the overhead placed on the network routers and the communication links, we propose that destinations do not leave the auxiliary groups after receiving the required retransmissions. Rather, a destination remains a member in the auxiliary groups for a period of time that is proportional to the error rate encountered by the system. Since retransmissions are split among the auxiliary groups, having several groups decreases the possibility that the destination receives another retransmission on the same group. Hence, we propose only one auxiliary group, to which all retransmissions are directed.

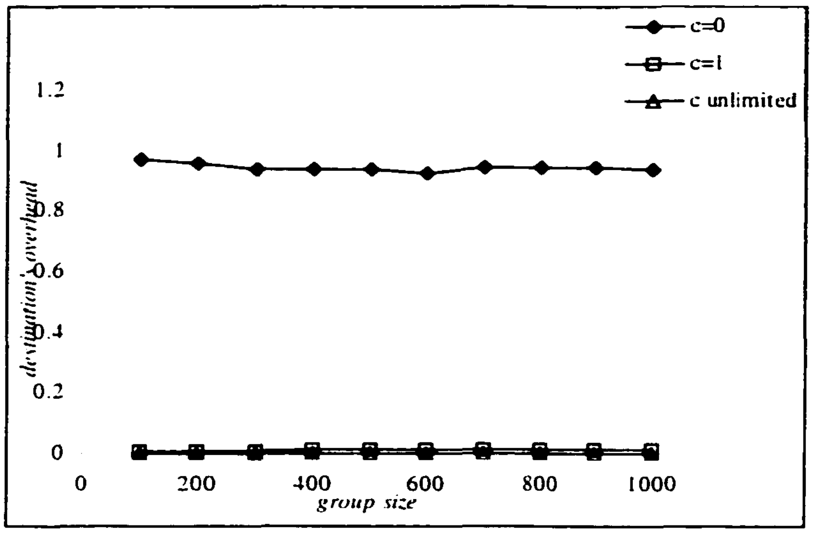

Fig. 50. Destinations' overhead in ARQ.

We have conducted a simulation study that compares our proposed system with the packet grouping system. We have considered two alternatives for routing techniques: broadcast and prune and shared tree. Our measurements include latency, destinations' overhead, and network bandwidth consumed. Fig. 49 gives the results of latency incurred in using multiple groups. The latency is a maximum when utilizing unlimited number of groups, and it decreases by allowing destinations to stay longer in the group. This applies for both CBT and broadcast and prune routing techniques. This result is intuitive since retransmissions can only be carried out after all interested destinations have joined the auxiliary group, and if all interested destinations are already in the group, retransmission can be performed immediately.

Fig. 50 shows the results for destinations overhead. As the figure shows, using unlimited number of groups eliminates destinations' overhead. The overhead is a maximum if there is no auxiliary group at all. However, using one group with destinations continue their memberships for a longer time gives a compromise between these two extremes, which trades latency against overhead. This result applies for both CBT and broadcast and prune routing techniques.

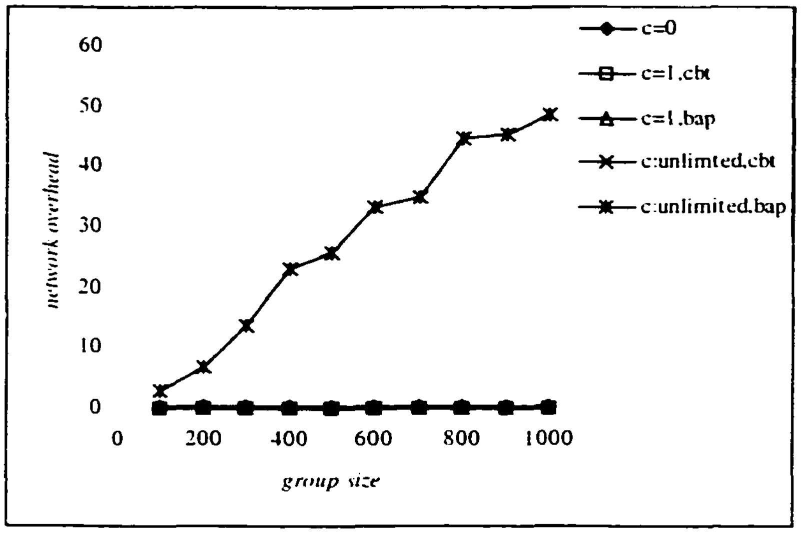

Fig. 51. Network overhead in ARQ.

Lastly, Fig. 51 gives results for network overhead in terms of bandwidth consumed. As the figure shows, the overhead in having unlimited number of groups in broadcast and prune network is too high. There are two sources of bandwidth overhead: delivering retransmissions to destinations unnecessarily and building and maintaining the auxiliary group. For broadcast and prune, the price of building and maintaining the delivery tree is very high, suggesting a slow membership dynamics (destinations continue their membership for longer periods). The main source of bandwidth overhead in CBT is in delivering unwanted retransmissions. In this case, a fast membership dynamics gives better results.

6.6 Multiple Multicast Groups in FEC

In this section, we propose utilizing multiple groups in multicast FEC. A main challenge in heterogeneous environment is that not all destinations experience the same error rate. Fixing the number of repair packets to one value for all group members either overwhelms the low error rate destinations with unneeded correction packets or delivers too few of them to the high error rate members. A solution to this problem is to group destinations according to the error rate they experience. For example, consider a multicast group consisting of n destinations. Assume that, for each 5 packets, x destinations experience 1 lost packet, y experience 2, and z have 3 errors. On the one hand, we can use one multicast group for all destinations. This approach requires sending 3 repair packets to all n destinations, wasting computational power at some destinations and network bandwidth along the path from the source to these destinations. On the other hand, we can have 2 auxiliary groups, the first carries 1 repair packet and the second carries 2 for every 5 data packets (see Fig. 52). In order to get the necessary and the sufficient amount of repair packets, the first x destinations join the first group, the other y destinations join the second group and the z destinations join both groups.

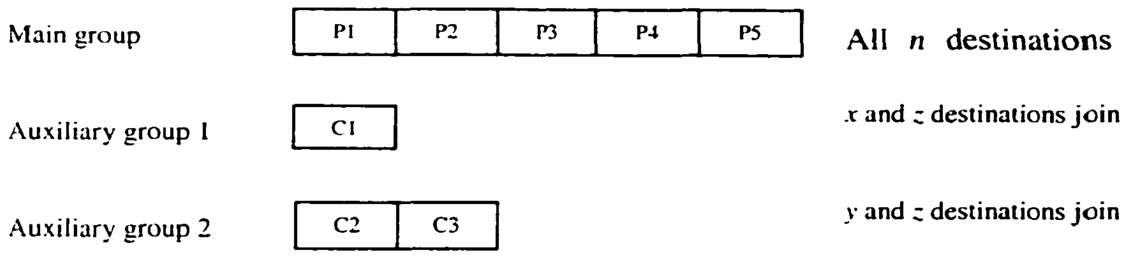

Fig. 52. Multiple Multicast groups in FEC.

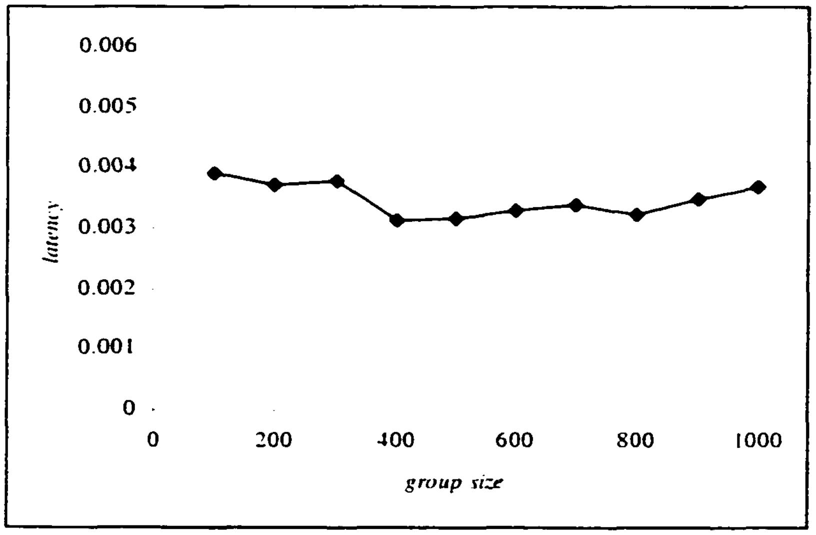

Fig. 53. Latency in FEC.

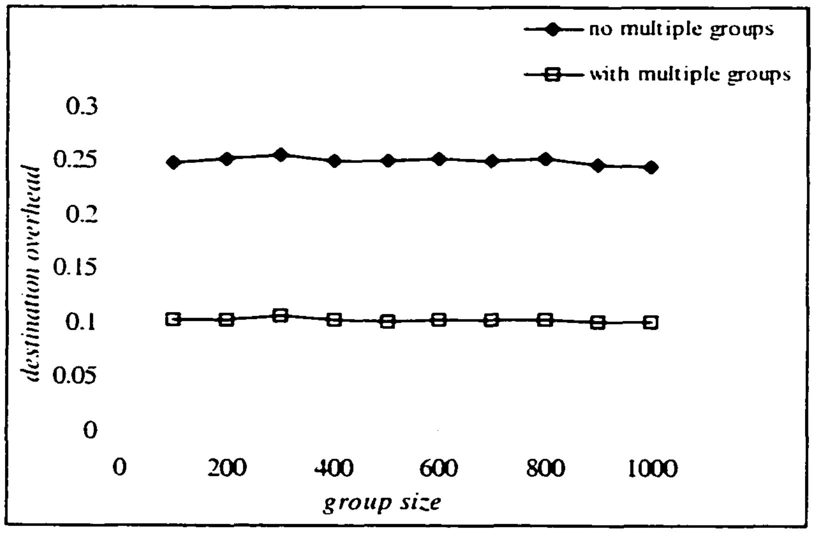

Fig. 54. Destinations' overhead in FEC

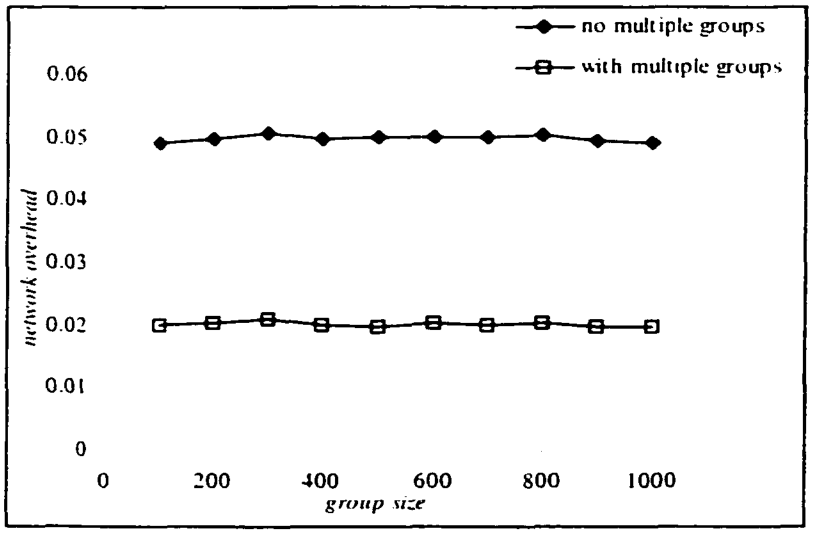

Fig. 55. Network Overhead in FEC.

We have conducted a simulation study the compares our proposed approach against the one that uses only one group for both data and codes for the same loss rate in both approaches. We have collected results for Latency, destinations' overhead, and network overhead in terms of wasted bandwidth in both CBT and broadcast and prune. Results shown are for CBT only. Similar results, not shown in figures, are obtained for broadcast and prune. Fig. 53 shows that the latency introduced as a result of using multiple groups (as correction codes may be delayed in order to give destinations a chance to adjust their membership in the communication groups) is negligible. gives the destinations' overhead in FEC. As the figure shows, a reduced overhead has been achieved in using multiple groups. Lastly, Fig. 55 shows the network bandwidth overhead in using multiple groups. As the figure shows, using multiple groups minimizes the network overhead.

6.7 Multiple Multicast Groups in Hybrid Techniques

Multiple groups can be utilized in hybrid techniques by combining the ideas introduced in ARQ and FEC. For hybrid techniques, all destinations are required to join the main group, which carries the original data. One or more auxiliary groups are dedicated for FEC repair codes. The number of these groups and the rate of code packets they carry depend on the error rates experienced and destinations join these groups based on their error rates. As in ARQ, there is only one auxiliary group for retransmissions. Destinations join this group to repair errors that could not be corrected using FEC. The duration for which a destination joins the retransmission group depends on the error rate it experiences.

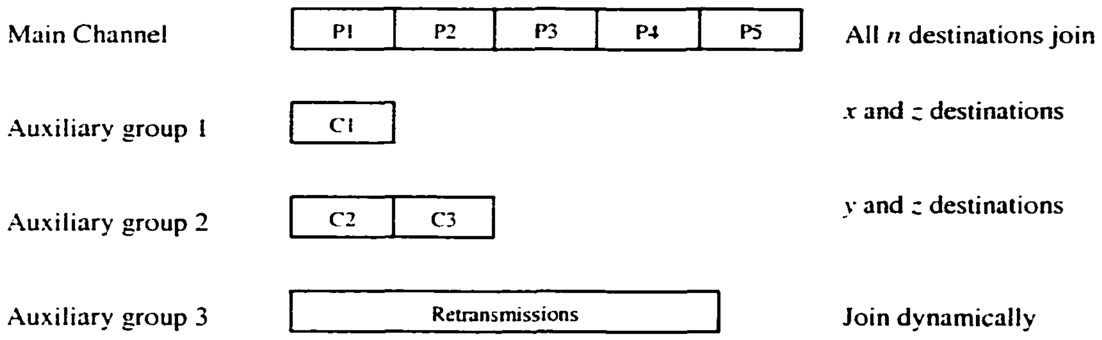

Fig. 56. Multiple multicast groups in hybrid techniques.

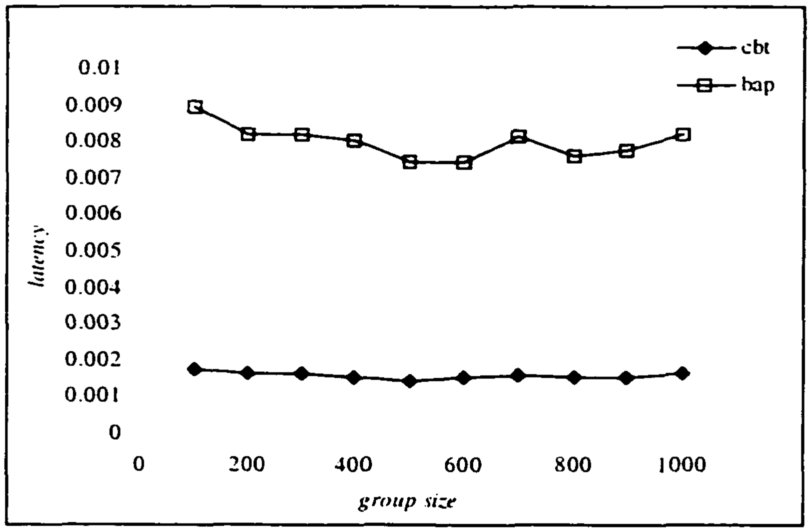

Fig. 57. Latency in the hybrid approach.

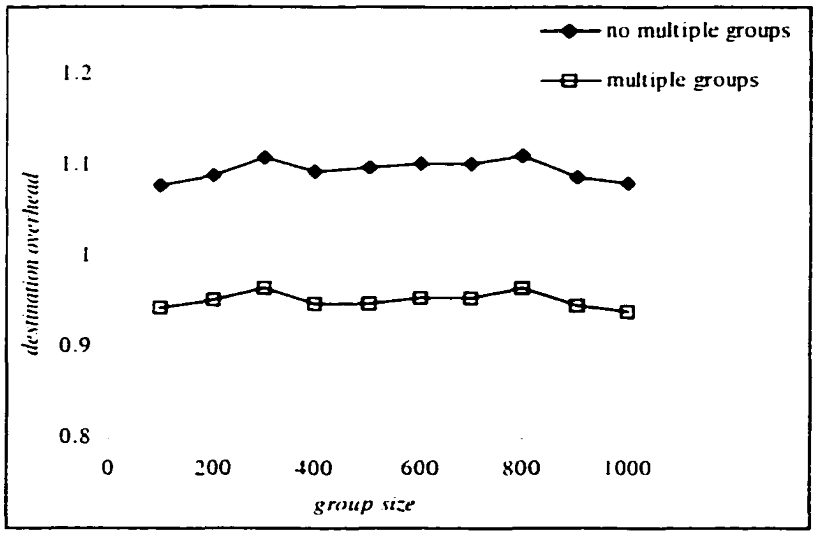

Fig. 58. Destinations' overhead in the hybrid approach.

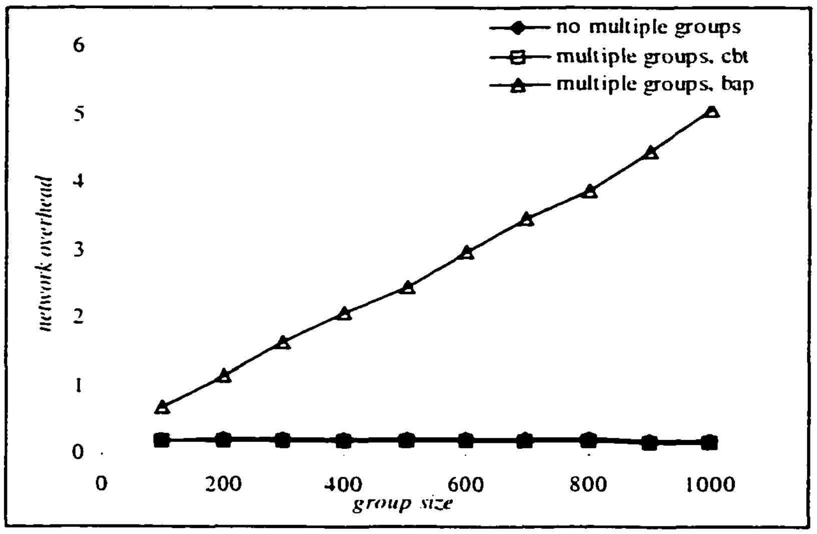

Fig. 59. Network overhead in the hybrid approach.

Fig. 56 gives an example of the technique. Much similar to the example given in Section 0, we assume the same number of destinations. In addition to the error rates given, we assume that occasionally some destinations may experience more errors, which requires retransmissions. Very similar to the ARQ situation, these destinations can join the retransmission group for a period of time that depends on the encountered error rate.

As with ARQ and FEC, we have conducted a simulation study for the hybrid approach. Fig. 57, Fig. 58, and Fig. 59 give the results for our simulations. As Fig. 57 shows, the latency introduced due the use of multiple groups is negligible, (we have found, although not shown in the figure, that latency can be reduced if destinations are allowed to maintain their membership in the retransmission group for longer time). Fig. 58 compares destinations' overhead using multiple groups with the approach not using them. As the figure shows, there is a reduction in overhead when using multiple groups. Fig. 59 shows the network overhead involved in using multiple groups. For broadcast and prune, network overhead is too high (since there is a high price in maintaining the deliver tree). The overhead in CBT, however, is comparable to that of the case of no multiple groups.

6.8 Multicast Error Control and Congestion and Flow Control

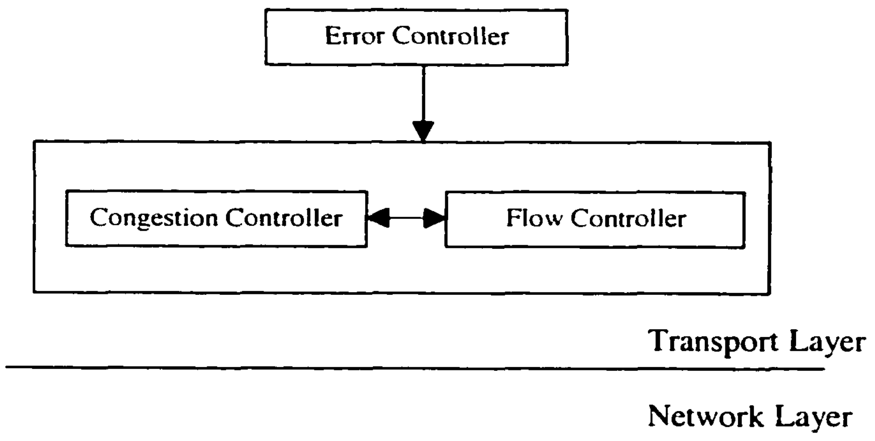

Fig. 60. Error control and congestion and flow control.

In this chapter and the previous one we have introduced separate techniques for end-to-end multicast error control and end-to-end multicast congestion and flow control. In both topics, we use multiple multicast groups to achieve the goal of each. It should be noted that using multiple multicast groups in these two modules is to achieve two different goals: to control flow and congestion on the one hand, and to control error on the other. However, the motivation behind using multiple multicast groups in both modules is the heterogeneity of the destinations and their network connections to the source.

Indeed, there is a strong relationship between the congestion and flow control and error control modules. In fact, as Fig. 60 shows, our error control module should be built on top of the congestion and flow control module. The congestion and flow control module deals with each multicast group produced by the error control module (whether a data group or auxiliary groups) individually. It regulates the traffic generated by each group independent of the other groups. The congestion and flow control module may further subgroup any of the error control module groups in order to control the network congestion or the flow between the source and the different destinations. That is, for every multicast group produced by the error control module, destinations are organized into subgroups and the transmission rate of each subgroup is controlled as has been discussed in the previous chapter.

6.9 Conclusion

In this chapter, we have studied the use of multiple groups to control multicast errors in heterogeneous environment. We have explored three approaches in multicast error control: ARQ, FEC, and hybrid. In a heterogeneous environment, different destinations experience different error patterns. In such an environment, using one group wastes resources at low error rate destinations and the network bandwidth along the paths from the source to these destinations. To overcome this problem, we have examined utilizing multiple groups in the three error control approaches.

The use of multiple groups in multicast ARQ has been previously introduced. We have enhanced the previous work in this area, taking into consideration the effect of network routing over having multiple groups. We have proposed two mechanisms in utilizing multiple groups in both FEC and hybrid approaches. We have compared the enhancement and our new techniques with existing ones. Our comparison is based on simulation studies. The results of the simulation show superiority of our techniques over the existing ones.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

7.1 Conclusion

Multimedia collaborative applications are gaining popularity as a solution that provides collaboration among several users. Multimedia collaborative applications exploit the computing and networking technologies along with the media acquisition and playback facilities to help a group of users, not necessarily residing in the same place, to work and interact together in a common task. This proves invaluable for many organizations such as large enterprises with sites located in different cities or even different countries. Examples of these applications include computer conferencing and distance learning

Essential to the success of this class of applications is an efficient multicast layer. In this dissertation, we investigate new designs for multicast services that better meet the requirements of multimedia collaborative applications. Typically, multimedia collaborative applications are of small group size, slow group membership dynamics, and awareness of participants' identities and locations. Moreover, they usually consist of several components such as audio, video, shared whiteboard, and single user application sharing engines that collectively help make the collaboration session successful. Each of these components has its characteristics and requirements from the communication layer that may differ from one component to another.

In this dissertation, we have identified the overall characteristics of multimedia collaborative applications and their individual components. We also have determined the service requirements of the various components from the communication layer. Based on our analysis, new techniques of multicast services that are more suitable for multimedia collaborative applications have been introduced. In particular, we have developed new designs for multicast address management and connection control, routing, congestion and flow control, and error control.

First, we have investigated multicast address management and port resolution in the Internet—which we have taken as an example of large inter-networks—and have developed new techniques for both problems. We have conducted an experimental work and have developed an analytical study to evaluate the effectiveness of our new techniques. For Multicast address management, we have classified multicast sessions into local sessions and global sessions. As for local sessions, we have presented three alternatives for managing the address space: central servers, partitioning over network's hosts, and detection and recovery. All these techniques have a negligible blocking probability. Partitioning the address space over the networks' hosts, however, results in the lowest latency and communication and processing overhead.

Managing the address space of global sessions with a central server may not be suitable because of the large latency and the bottlenecks around the server. Also using detection and recovery implies restarting some multicast sessions after detecting collisions, which may not be appropriate in many situations. Partitioning the address space over the networks incurs large blocking probability but minimizes latency and avoids communication and processing bottlenecks. We feel it is better to leave each network the choice of managing its address space, whether using a central server or partitioning the address space over its hosts.

For port resolution, we have presented a technique to reserve a small portion of the host's port space to the multicast communication. Knowing that port requests mostly are for unicast communication, the blocking probability in multicast ports is negligible. This technique does not eliminate the port blocking problem and it requires system support to reserve part of the port space for multicast. It, however, reduces the probability of port blocking to an insignificant level. Also, the system load is much smaller than that of having virtual ports—which also requires a system support and maintains a mapping table.

Second, we have exploited the problem of multicast routing and we have developed a new routing technique that combines the advantages of broadcast and prune and core based tree. Specifically, our technique produces the minimum delay tree—which is required for audio and video streams—while the overhead in building and maintaining the multicast tree is minimal. Basic assumptions for our technique to work are small group size and slow membership dynamics, which are typical in most collaboration sessions. Our technique relies on advertising a rendezvous point (RVP) where the destinations and sources can meet. Our algorithm works in three steps. First, destinations and sources are required to send membership messages towards the RVP. Then, the RVP multicasts its information to all sources. Last, sources use the received membership to build the multicast trees. We have studied the different aspects in our technique and have modeled it using flow charts for the various involved entities. We also have introduced three variations of our technique to cope with the different cases of multicast collaborative applications. To evaluate our work, we have presented a simulation study that compares the new technique with broadcast and prune and core based routing. Results of the simulation demonstrate the superiority of the new technique over the other two approaches.

Third, we have explored the problems of congestion and flow control in multicast communication and we have presented a new technique that is based on using multiple multicast groups to solve both problems. The new technique is window based that requires the source to maintain a window per group. To avoid overloading the source with destinations' feedback, a representative per group is designated to collect the feedback from the rest of the group members and relay the feedback to the source. We have presented solutions to allow destinations to migrate between groups and to permit group merging and splitting. To evaluate our approach, we have presented an analytical study that compares the new technique with the cases of going with the slowest destination and the no control mechanism in real time communications where retransmissions are not allowed. Results of our study shows that our technique outperforms the other two approaches.

Last, we have examined multicast error control. We have investigated three approaches in multicast error control: ARQ, FEC, and hybrid. In a heterogeneous environment different destinations experience different error patterns. Thus, using one group wastes resources at low error rate destinations and the network bandwidth along the paths from the source to these destinations. To overcome this problem, we have examined utilizing multiple groups in the three error control approaches. First, We have enhanced a previous work in ARQ, taking into consideration the effect of network routing over having multiple groups. Second, we also have proposed two mechanisms in utilizing multiple groups in both FEC and hybrid approaches. We have compared the enhancement and our new techniques with existing ones using simulation. The results of the simulation show superiority of our techniques over the existing ones.

7.2 Future work

As many research efforts, this dissertation ends with more questions than what it has started with. Providing the suitable multicast services for multimedia collaborative applications is gaining the attention of many researchers. In this dissertation, we have investigated address management and connection control, routing, congestion and flow control, and error control. Many of the multimedia collaborative applications, however, require a security scheme where data can be delivered safely to the session participants. Further work is needed to provide a secure multicast communication that meets the requirements for multimedia collaborative applications.

We have presented multicast address management schemes for local and global sessions. Assigning addresses for global sessions, however, faces high blocking probability, since the address space is partitioned over many domains. The requirement is to have an efficient address management scheme that prevents collision and minimizes blocking at the same time. An approach to this problem is to have a multicast address management authority that leases multicast addresses to requesting domains. Related to this approach is how to secure the lease process and prevent others from using addresses they have not leased. Further effort is needed to investigate this approach.

In our routing technique, we assume a rendezvous point (RVP) where destinations and sources can meet. The selection of the RVP plays a major role in the overhead involved in building the multicast tree and in the join latency experienced by the destinations and the sources. In this dissertation, we have suggested to select one of the sources (the first one to send to the group for example) as the RVP. Being simple, and since we assume a low overhead in building and maintaining the tree, selecting one of the sources as the RVP seems attractive. A more concrete study, however, is required to validate our assumptions. We also have introduced three alternatives beside our new technique for multicast routing. Each of these alternatives performs well in certain situations. In this dissertation, we have introduced a qualitative comparison between them. Required is a quantitative study that determines which alternative is best for what situation.

We have presented an end-to-end technique to solve the problem of congestion and flow control in multicast communication. In our evaluation study, we have analytically measured the throughput for real time applications where there are no retransmissions. For reliable transmissions (using retransmissions), there are two main schemes: go back n, and selective repeat. More work is needed to evaluate the performance of our technique in these cases. Another measure that can be used to evaluate the various approaches is the delay experienced by the destinations.

Last, we have introduced a solution that uses multiple multicast groups for error control. We also have presented a brief discussion of how the error control module can be integrated with the congestion and flow control task. However, the different groups of the error control module may have different urgencies in packet delivery. More effort is required to determine how to prioritize the different error control module groups to the congestion and flow control module.

REFERENCES

[1] H. Abdel-Wahab and M. Feit, "XTV: A Framework for Sharing X Window Clients in Remote Synchronous Collaboration", *Proc. IEEE TriComm '91: Communications for Distributed Applications & Systems*, Chapel Hill, NC, USA, pp. 159-167, 1991.

[2] H. Abdel-Wahab, O. Kim, P. Kabore, and J.P. Favreau, "Java-based Multimedia Collaboration and Application Sharing Environment," *Proc. of the Colloque Francophone sur l'Ingenierie des Protocoles (CFIP '99)*, Nancy, France, April 26-29, 1999.

[3] H. Abdel-Wahab, A. Youssef, and K. Maly, "Distributed Management of Exclusive Resources In Collaborative Multimedia Systems," *The Third IEEE Symposium on Computers and Communications (ISCC'98)*, Athens, Greece, pp. 115-119, 1998.

[4] A. Al-Shammari and A. Karmouch, "On-demand Multimedia Courseware Delivery over the Network," *Proc. INDC'98 7th IFIP/ICCC Conference on Information Networks and Data Communications*, Portugal, 1998.

[5] E. Altman, T. Basar, and R. Srikant, "Congestion Control as a Stochastic Control Problem with Action Delays," *Automatica*, vol. 35, no. 12, pp. 1937-1950, December 1999.

[6] S. Armstrong, A. Freier, and K. Marzullo, "Multicast Transport Protocol", Internet RFC 1301, February 1992.

[7] J. Atwood, O. Catrina, J. Fenton, and T. Strayer, "Reliable Multicasting in the Xpress Transport Protocol," *Proc. of the 21st Local Computer Networks Conference*, Minneapolis, MN, USA, pp. 202-211, October 13-16, 1996.

[8] A. Ballardie, "A New Approach to Multicast Communication in a Datagram Internetwork," PhD thesis, University of College London, May 1995.

[9] A. Ballardie, P. Francis, and J. Crowcroft, "Core Based Trees (CBT) An Architecture for Scalable Inter-Domain Multicast Routing," *Computer communication review*, vol. 23, no. 4, pp. 85-95, October 1993.

[10] S. Bhattacharyya, J. Kurose, D. Towsley, R. Nagarajan, "Efficient Rate-Controlled Bulk Data Transfer using Multiple Multicast Groups," *Proc. IEEE Infocom'98*, San Francisco, CA, USA, April 1998.

[11] S. Bhattacharyya, D. Towsley, and J. Kurose, "The Loss Path Multiplicity Problem in Multicast Congestion Control," *Proc. IEEE Infocom'99*, New York, NY, USA, March 1999.

[12] K. Birman, R. Cooper, and B. Gleeson, "Design Alternatives for Process Group Membership and Multicast," Tech. Rep. TR91-1257, Cornell University, December 1991.

[13] J. Blomer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, "An XOR-Based Erasure-Resilient Coding Scheme," Tech. Rep. ICSI TR-95-048, International Computer Sciences Institute, August 1995.

[14] R. Braudes and S. Zabele, "Requirements for Multicast Protocols," Internet RFC 1458, May 1993.

[15] R. Cigno and M. Gerla, "Modeling Window Based Congestion Control Protocols with Many Flows," *Performance evaluation*, vol. 36, no. 1, pp. 289-306, August 1999.

[16] Y. Dalal and R. Metcalfe, "Reverse Path Forwarding of Broadcast Packets," *Communications of the ACM*, vol. 21, No. 12, pp. 1040-1048, December 1978.

[17] S. Deering, "Host Extensions for IP Multicast," Internet RFC 1112, August 1997.

[18] S. Deering, "Multicast Routing in a Datagram Internetwork," PhD thesis, Stanford University, December 1991.

[19] S. Deering, "Multicast Routing in Internetworks and Extended LANs," *Proc. SIGCOMM'88*, Stanford, CA, USA, vol. 18, no. 4, August 1988.

[20] S. Deering, D. Estrin, D. Farinacci, J. Jacobson, C-G. Liu, and L. Wei, "The PIM Architecture for Wide-area Multicast Routing," *IEEE/ACM transactions on networking*, vol. 4, no. 2, pp. 153-162, April 1996.

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

[22] E. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269-271, 1959.

[23] C. Diot, W. Dabbous, and J. Crowcroft, "Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms," *IEEE journal on selected areas in communications*, vol. 15, no. 3, pp. 277-290, April 1997.

[24] A. Eleftheriadis, S. Pejhan, and D. Anastassiou, "Address Management and Connection Control for Multicast Communication Applications," *Proc. IEEE Infocom'95*, Boston, MA, USA, pp. 386-393, April 1995.

[25] C. Ellis, S. Gibbs, and G. Rein, "Groupware: Some Issues and Experiences," *Communications of the ACM*, vol. 34, no. 1, pp. 39-58, January 1991.

[26] S. Floyd and K. Fall, "Promoting the Use of End-to-end Congestion Control in the Internet," *IEEE/ACM transactions on networking*, vol. 7, no. 4, pp. 458-472, August 1999.

[27] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," *IEEE/ACM transactions on networking*, vol. 5, no. 6, pp. 784-803, December 1997.

[28] V. Fuller, T. Li, J. Yu, and K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy," Internet RFC 1519, September 1993.

[29] M. Gerla and L. Kleinrock, "Flow Control: A Comparative Survey," *IEEE Transactions on Communications*, vol. COM-28, no. 4, pp. 553-574, April 1980.

[30] S. Golestani and K. Sabnani, "Fundamental Observations on Multicast Congestion Control in the Internet," *Proc. Infocom'99*, New York, NY, USA, March 1999.

[31] I. Greif, ed., "Computer Supported Cooperative Work: a book of readings," Morgan Kaufmann, 1988.

[32] H. Hofte, "Working Apart Together: Foundations for Component Groupware," PhD thesis, University of Twente, the Netherlands, June 1998.

[33] H. Holbrook, S. Singhal, and D. Cheriton, "Log-Based Receiver Reliable Multicast for Distributed Interactive Simulation," *Proc. ACM SIGCOMM'95*, Cambridge, MA, USA, pp. 328-341, August 1995.

[34] J. Huang, C. Yang, and N. Fang, "A Novel Congestion Control Mechanism for Multicast Real-time Connections," *Computer communications*, vol. 22, no. 1, pp. 56-72, January 1999.

[35] C. Huitema, "The case for packet level FEC," *Proc. of IFIP 5th International Workshop on Protocols for High Speed Networks (PfHSN'96)*, INRIA, Sophia Antipolis, FRANCE, October 1996.

[36] F. Hwang and D. Richards, "Steiner Tree Problem," *Networks*, vol. 22, no. 1, pp. 55-89, January 1992.

[37] V. Jacobson, "Congestion Avoidance and Control," *Proc. ACM SIGCOMM'88*, Stanford, CA, USA, vol. 18, no. 4, August 1988.

[38] V. Jacobson and S. McCanne, "vat - LBL Audio Conferencing Tool," URL: http://www-nrg.ee.lbl.gov/vat/.

[39] R. Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling," John Wiley & Sons, 1991.

[40] R. Jain, "Congestion Control in Computer Networks: Issues and Trends," *IEEE Network*, vol. 4 no. 3, pp. 24-30, May 1990.

[41] H. Kanakia, P. Mishra, and A. Reibman, "An Adaptive Congestion Control Scheme for Real Time Packet Video Tranport," *IEEE/ACM transactions on networking*, vol. 3, no. 6, pp. 671-682, December 1995.

[42] R. Karp, "Reducibility among Combinatorial Problems," *in Complexity of Computer Communications*, R. Miller and J. Thatcher (Eds.) Plenum Press, New York, pp. 85-103, 1972.

[43] R. Karp, "The Probabilistic Analysis of Some Combinatorial Search Algorithms," *in J. Traub (Ed.) Algorithms and Complexity: New Directions and Recent Results*, Academic Press, New York, pp. 1-19, 1976.

[44] S. Kasera, J. Kurose, and D. Towsley, "A Comparison of Server-Based and Receiver-Based Local Recovery Approaches for Scalable Reliable Multicast," *Proc. IEEE Infocom'98*, San Francisco, CA, USA, April 1998.

[45] S. Kasera, J. Kurose, and D. Towsley, "Scalable Reliable Multicast using Multiple Multicast Groups," *Proc. ACM Sigmetrics'97*, Seattle, WA, USA, pp. 64-74, June 1997.

[46] R. Kermode, "Scoped Hybrid Automatic Repeat Request with Forward Error Correction (SHARQFEC)," *Proc. ACM Sigcomm'98*, Vancouver, Canada, pp. 268-277, September 1998.

[47] S. Keshav, "A Control-Theoretic Approach to Flow Control," *Proc. ACM Sigcomm'91*, Zurich, Switzerland, pp. 3-15, September 1991.

[48] A. Koifman and S. Zabele, "RAMP: A Reliable Adaptive Multicast Protocol," *Proc. IEEE Infocom'96*, San Francisco, CA, USA, March 1996.

[49] A. Law and W. Kelton, "Simulation Modeling and Analysis," McGraw-Hill, 1991.

[50] L. Lehman, S. Garland, and D. Tennenhouse, "Active Reliable Multicast," *Proc. IEEE Infocom'98*, San Francisco, CA, USA, March 1998.

[51] M. Leland, R. Fish, and R. Kraut, "Collaborative Document Production Using Quilt," *Proc. ACM CSCW'88*, Prtland, OR, USA, pp. 206-215, September 1988.

[52] K. Maly, H. Abdel-Wahab, C. Overstreet, C. Wild, A. Gupta, A. Youssef, E. Stoica and E. Al-Shaer, "Distance Learning and Training over Intranets", *IEEE Internet Computing*, vol 1, no. 1, pp. 60-71, 1997.

[53] S. McCanne and V. Jacobson, "vic: A Flexible Framework for Packet Video," *ACM Multimedia'95*, San Francisco, CA, USA, pp. 511-522, November 1995.

[54] Microsoft NetMeeting, URL://http:www.microsoft.com/netmeeting/

[55] P. Mishra, H. kanakia, and S. Tripathi, "On Hop-by-Hop Rate-Based Congestion Control," *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, pp. 224-239, April 1996.

[56] W. Mostafa and M. Singhal, "A Taxonomy of Multicast Protocols for Internet Applications," *Computer Communications*, vol. 20, no. 16, pp. 1448-1457, January 1998.

[57] J. Moy, "Multicast Routing Extensions for OSPF," *Communications of the ACM*, vol. 37, no. 8, pp. 61-66, August 1994.

[58] J. Nagle, "Congestion Control in IP/TCP Internetworks," *Computer Communication Review*, vol. 25, no. 1, January 1995.

[59] J. Nonnenmacher, E. Biersack, and D. Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 349-361, August 1998.

[60] J. Nonnenmacher, M. Lacher, M. Jung, and E. Biersack, "How bad is Reliable Multicast without Local Recovery?" *Proc. IEEE Infocom'98*, San Francisco, CA, USA, March 1998.

[61] G. Pal and S. Agrawal, "Window-Based Congestion Control in a Packet Switched Network with Voice and Data Transmission," *Computer Communications*, vol. 19, no. 6-7, pp. 612-618, June 1996.

[62] J. Pansiot and D. Grad, "On Routes and Multicast Trees in the Internet," *Computer Communication Review*, vol. 28, no. 1, January 1998.

[63] S. Paul, K. Sabnani, J. Lin, and S. Bhattacharyya, "Reliable Multicast Transport Protocol (RMTP)," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 407-421, April 1997.

[64] S. Pejhan, A. Eleftheriadis, and D. Anastassiou, "Distributed Multicast Address Management in the Global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1445-1456, October 1995.

[65] M. Podolsky, C. Romer, and S. McCanne, "Simulation of FEC-Based Error Control for Packet Audio on the Internet," *Proc. Infocom'98*, San Francisco, CA, USA, March 1998.

[66] G. Poo and A. Goscinski, "Performance Comparison of Sender-Based and Receiver-Based Reliable Multicast Protocols," *Computer Communications*, vol. 21, no. 7, pp. 597-605, June 1998.

[67] L. Postel, "Internet Protocol," Internet RFC 791, September 1981.

[68] W. Reinhard, J. Schweitzer, G. Volksen, and M. Weber, "CSCW Tools: Concepts and Architectures," *IEEE Computer*, vol. 27, no. 5, pp. 28-36, May 1994.

[69] J. Reynolds and J. Postel, "Assigned Numbers," Internet RFC 1700, October 1994.

[70] I. Rhee, N. Balaguru, and G. Rouskas, "MTCP: Scalable TCP-like Congestion Control for Reliable Multicast," *Proc. Infocom'99*, New York, NY, USA, March 1999.

[71] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols," *Computer Communication Review*, vol. 27, no. 2, April 1997.

[72] D. Rubenstein, S. Kasera, D. Towsley, and J. Kurose, "Improving Reliable Multicast Using Active Parity Encoding Services (APES)," *Proc. IEEE Infocom'99*, New York, NY, USA, March 1999.

[73] D. Rubenstein, J. Kurose, and D. Towsley, "Real-Time Reliable Multicast Using Proactive Forward Error Correction," *Proc. IEEE NOSSDAV'98*, Cambridge, UK, July 1998.

[74] M. Schwartz, "Telecommunication Networks: Protocols, Modeling and Analysis," Addison-Wesley, 1987.

[75] A. Shaikh and K. Shin, "Destination-Driven Routing for Low-Cost Multicast," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 373-381, April 1997.

[76] M. Stefik, G. Foster, D.G. Bobrow, K. Kahn, S. Lanning, and L. Suchman, "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings," *Communications of the ACM*, vol. 30, no. 1, pp. 32-47, January 1987.

[77] E. Stoica, "Multiple Streams Synchronization in Collaborative Multimedia Systems," Ph.D. Dissertation, Computer Science Department, Old Dominion University, July 1998.

[78] A. Thyagarajan and S. Deering, "Hierarchical Distance Vector Multicast Routing for the MBone," *Proc. ACM Sigcomm'95*, Cambridge, MA, USA, pp. 60-66, August 1995.

[79] D. Towsley, J. Kurose, and S. Pingali, "A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 398-406, April 1997.

[80] J. Turner, "New Directions in Communications (or Which Way to the Information Age)," *IEEE Communication Magazine*, vol. 24, pp. 8-15, October 1986.

[81] N. Venkitaraman, T. Kim, K. Lee, S. Lu, and V. Bharghavan, "Design and Evaluation of Congestion Control Algorithms in the Future Inernet," *Proc. Sigmetrics'99*, Atlanta, GA, USA, pp. 212-213, May 1999.

[82] L. Vicisano, L. Rizzo, J. Crowcroft, "TCP-like Congestion Control for Layered Multicast Data Transfer," *Proc. IEEE Infocom'98, San Francisco*, CA, USA, April 1998.

[83] D. Waitzman, C. Patridge, and S. Deering, "Distance Vector Multicast Routing Protocol," Internet RFC 1075, November 1988.

[84] M. Yamamoto, J. Kurose, D. Towsley, and H. Ikeda, "A Delay Analysis of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols," *Proc. IEEE Infocom'97*, Kobe, Japan, April 97.

[85] C. Yang and A. Reddy, "A Taxonomy for Congestion Control Algorithms in Packet Switching Networks," *IEEE Network*, vol. 9, no. 4, pp. 34-45, July 1995.

[86] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," *Proc. IEEE Infocom'96*, San Francisco, CA, USA, March 1996.

VITA

Emad Mohamed was born in Cairo, Egypt. He received his Bachelor of Science in Electronics and Communications Engineering from Faculty of Engineering, Cairo University, Cairo, Egypt, in August 1988. He worked as Lecturer for the Department of Computer Science of Institute of Statistical Studies and Research, Cairo University, from 1989 to 1994 and earned a Master of Science in Computer Engineering in 1994 from the same university. In Fall 1994, Emad joined University of Nebraska at Omaha and received a Master of Science in Computer Science in 1996. In August 1996, he joined the Department of Computer Science at Old Dominion University, Norfolk, Virginia as a Ph.D. student. Emad will resume his faculty position at Cairo University after his dissertation defense.