SCALABILITY, THROUGHPUT STABILITY AND EFFICIENT BUFFERING IN RELIABLE MULTICAST PROTOCOLS[°]

A Dissertation

Presented to the Faculty of the Graduate School

of Ege University, Izmir, Turkey

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Oznur Ozkasap

May 2000

[°] This dissertation research has been conducted at Department of Computer Science, Cornell University; supervised by Professor Kenneth P. Birman, and was partially supported by a TUBITAK (Turkish Scientific and Technical Research Council)-NATO grant.

ABSTRACT

SCALABILITY, THROUGHPUT STABILITY AND EFFICIENT BUFFERING IN RELIABLE MULTICAST PROTOCOLS

Oznur Ozkasap*

This study investigates the issues of scalability, throughput stability and efficient buffering in reliable multicast protocols. The focus is on a new class of scalable reliable multicast protocol, Pbcast that is based on an epidemic loss recovery mechanism. The protocol offers scalability, throughput stability and a bimodal delivery guarantee as the key features. A theoretical analysis study for the protocol is already available.

This thesis models Pbcast protocol, analyzes the protocol behavior and compares it with multicast protocols offering different reliability models, in both real and simulated network settings. Techniques proposed for efficient loss recovery and buffering are designed and implemented on the simulation platform as well. Extensive analysis studies are conducted for investigating protocol properties in practice and comparing it with other classes of reliable multicast protocols across various network characteristics and application scenarios. The underlying network for our experimental model is the IBM SP2 system of the Cornell Theory Center. In the simulation model, we used the ns-2 network simulator as the underlying structure. Performance metrics, such as scalability, throughput stability, link utilization and message latency distribution, are analyzed. It is demonstrated that Pbcast protocol scales well, and in contrast to the other scalable reliable multicast protocols, it gives predictable reliability even under highly perturbed conditions.

* Current contact info: Assistant Professor, College of Engineering, Koc University, Sariyer, Istanbul, Turkey. E-mail: oozkasap@ku.edu.tr

LIST OF ABBREVIATIONS

ACK	Acknowledgement
ALF	Application Level Framing
CBR	Constant Bit Rate
CBT	Core Based Tree
CRC	Cyclic Redundancy Check
DVMRP	Distance Vector Multicast Routing Protocol
FIFO	First In First Out
IP	Internet Protocol
LAN	Local Area Network
MFTP	Multicast File Transfer Protocol
NAK	Negative Acknowledgement
OTERS	On-Tree Efficient Recovery using Subcasting
PGM	Pragmatic General Multicast
PIM	Protocol Independent Multicast
RMTP	Reliable Message Transfer Protocol
RPC	Remote Procedure Call
SRM	Scalable Reliable Multicast
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
XTP	Xpress Transfer Protocol

TABLE OF CONTENTS

1. Introduction

The availability of high speed networks and the growth of the Internet have triggered the use of multicast communication in large scale settings. Furthermore, the widespread availability of IP multicast (Deering and Cheriton, 1990) and the Mbone (Kumar, 1995) have important consequences in terms of the use of large-scale multicast communication. These developments have considerably increased both the geographic extent and the size of communication groups. Distributed applications such as Internet media distribution, electronic stock exchange, computer supported collaborative work, air traffic control and reliable information dissemination need to distribute data among multiple participants. As the size and geographic extent of such applications increase, scalable reliable multicast protocols become an essential underlying communication structure.

Several large-scale distributed applications exploiting multicast communication require reliable delivery of data to all participants. In addition, scalability, throughput stability, efficient loss recovery and buffer management are essential communication properties in large-scale settings.

There are two primary classes of multicast protocols offering reliability guarantees. One class of protocols offers strong reliability guarantees such as atomicity, delivery ordering, virtual synchrony, real-time support, security properties and network-partitioning support. The other class offers support for best-effort reliability in large-scale settings.

Although protocols providing strong reliability guarantees are useful for many applications, they have some limitations in terms of scalability and throughput stability.

The drawback is that in order to obtain strong reliability guarantees, costly protocols are used and the possibility of unstable or unpredictable performance under failure scenarios is accepted. These protocols allow limited scalability. As mentioned in (Piantoni and Stancescu, 1997) the maximum number of participants must not exceed about fifty to one hundred. Otherwise, transient performance problems can cause these protocols to exhibit degraded throughput.

The second class of protocols offers support for best-effort reliability in large-scale settings. These protocols overcome message loss and failures, but they do not guarantee end-to-end reliability. For instance, group members may not have a consistent knowledge of group membership, or a member may leave the group without informing the others. This class of protocols is suitable for large-scale networks and they do scale beyond the limits of protocols offering strong reliability guarantees. When the message loss probability is very low or uncommon, they can give a very high degree of reliability. But, failure scenarios such as router overload and system-wide noise which are known to be common in Internet protocols can cause these protocols to behave pathologically (Labovitz et al., 1997; Paxson, 1997).

For large-scale applications such as Internet media distribution, electronic stock exchange and distribution of flight telemetry data in air traffic control systems, the throughput stability guarantee is extremely important. This property entails the steady delivery of multicast data stream to correct destinations. Throughput instability problem applies to both classes of reliable multicast protocols that we discussed.

Buffering scalability is another important issue for large-scale distributed applications that motivate our work. Very little attention has been paid to solve the buffer management problem in scalable reliable multicast protocols. Most existing protocols either ignore the problem, or provide only an ad hoc solution.

This thesis study focuses on a new option in scalable reliable multicast protocols. We call this protocol bimodal multicast, or Pbcast (probabilistic multicast) in short (Birman et al., 1999). The behavior of Pbcast can be predicted given simple information on how processes and the network behave most of the time. The protocol exhibits stable

throughput under failure scenarios that are common on real large-scale networks. In contrast, this kind of behavior can cause other reliable multicast protocols to exhibit unstable throughput.

This study investigates the issues of scalability, throughput stability and efficient buffering in reliable multicast protocols. We developed experimental and simulation models for Pbcast protocol. The underlying network for our experimental model is the IBM SP2 system of the Cornell Theory Center. In the experimental model, we accomplished an analysis study for investigating the behavior and evaluating the performance of Pbcast, and comparing it with protocols offering strong reliability guarantees. For this purpose, we designed and constructed several group communication application scenarios. In the simulation model, we used the ns-2 (Bajaj et al., 1999) network simulator as the underlying structure. We designed and implemented basic Pbcast protocol on top of ns-2. In addition, for fast error recovery, we developed and modeled some optimizations to the protocol, and also used the simulation model of a scalable reliable multicast protocol for comparison across various network characteristics and application scenarios. By using the simulation model, we performed extensive simulation studies for investigating several issues that are important for scalable reliable multicast protocols. We analyzed performance metrics such as scalability, throughput stability, link utilization and message latency distribution for both Pbcast and a reliable multicast protocol offering best-effort reliability.

We demonstrate that Pbcast protocol scales well, and in contrast to the other scalable reliable multicast protocols it gives predictable reliability even under highly perturbed conditions. We include a variety of results demonstrating the throughput instability problem in existing multicast protocols based on different reliability models.

We also implement some techniques for buffering scalability in reliable multicast protocols, and demonstrate the efficiency of them by extensive simulations.

Contributions accomplished in this thesis study can be described as follows. This study models Pbcast protocol, analyzes the protocol behavior and compares it with multicast protocols offering different reliability models, in both real and simulated network

settings. First, an experimental model for Pbcast was developed, and several group communication applications were constructed for investigating protocol properties in real network settings. In addition, a comparison study with protocols offering strong reliability guarantees has been accomplished under the same network settings. Next, a simulation model for Pbcast was developed. In the simulation model, design and implementation of basic Pbcast have been accomplished. Furthermore, for fast error recovery, some optimizations to the protocol were developed. In contrast to the experimental model, simulation methods made possible to evaluate protocol's performance on several network topologies, failure models and large scale settings. Furthermore, a comparison study with a well-known scalable reliable multicast protocol offering best-effort reliability has been accomplished. In this thesis study, extensive analysis studies evaluating the scalability and stability metrics of the protocols for both experimental and simulation results have been performed. This thesis study also describes a technique for efficient buffering in reliable multicast protocols. The idea was first suggested by Robbert van Renesse, and in the simulation model accomplished in this thesis study, the technique has been integrated to the Pbcast protocol. Then, a simulation and analysis study, for validating the effectiveness of the technique, has been conducted.

The dissertation is organized as follows. Chapter 2 provides background for reliable multicast protocols, explains the throughput stability concept, investigates the buffering issue in the context of reliable multicast protocols, and provides motivation and application classes that this thesis study focuses on. Chapter 3 starts by giving information on the epidemic communication and then describes the Pbcast protocol in detail. Chapter 4 gives details of the experimental model, results and analysis. Chapter 5 gives details of the simulation model, protocol design and implementations. Chapter 6 first describes network and application characteristics of our simulation study. Then, it explains simulation studies, results and analysis in detail. Chapter 7 first describes the technique for efficient buffering. Then, it gives the details of the simulation study, results and analysis of the technique. Chapter 8 is the conclusion.

2. Background

Multicast is an important communication paradigm for constructing distributed computing applications. Basically, it is a way of transmitting a message to the members of a specified group of processes. The abstraction of a group is a logical name for a set of processes whose membership may change with time. Many different types of entities can be considered as group members such as processes, processors, name servers, database servers and sub-networks of a large-scale communication system. Groups are mainly used in distributed systems for distributing information and work, replicating data, naming and monitoring (Couloris et al., 1994; Mullender 1993). The key property of a process group is that when a message is sent to the group, all correct members need to receive that message. This is a type of one-to-many communication called multicast where there exists one sender and many receivers.

The first system in the literature introducing support for group communication was the V system (Cheriton and Zwaenepoel, 1985). The system offered a best-effort multicast mechanism as an operating system primitive, but lacked guarantees for reliable or ordered delivery of messages.

Several distributed applications exploiting multicast communication require reliable delivery of messages to all destinations. Therefore, a reliable multicast protocol is the basic building block of such an application. Example systems making use of reliable multicast protocols include electronic stock exchanges, air traffic control systems, health care systems, and factory automation systems. The degree of reliability guarantees

required by such applications differs from one setting to another. Thus, reliability guarantees provided by multicast communication protocols split them into two broad classes. One class of protocols offers strong reliability guarantees such as atomicity, delivery ordering, virtual synchrony, real-time support, security properties and network-partitioning support. The other class offers support for best-effort reliability in large-scale settings.

2.1 Strong Reliability Guarantees

One of the key properties provided by a reliable multicast protocol is atomicity. Informally, this means that a multicast message is either received by all destinations that do not fail or by none of them. Atomicity, which is also called all-or-nothing delivery, is a useful property, because a process that delivers an atomic multicast knows that all the operational destinations will also deliver the same message. This guarantees consistency with the actions taken by group members (Cristian et al., 1985).

Some applications also require ordering during the delivery of messages. Ordered multicast protocols ensure that the order of messages delivered is the same on each operational destination (Hadzilacos and Toueg, 1993). Different forms of ordering are possible such as FIFO, causal and total ordering. The strongest form among these is the total order guarantee that ensures that multicast messages reach all of the members in the same order (Lamport, 1978).

Distributed real-time and control applications need timing support in reliable multicast protocols. In these systems, multicast messages must be delivered at each destination by their deadlines.

The virtual synchrony model (Birman and Joseph, 1987) was introduced in the Isis system. In addition to message ordering, this model guarantees that membership changes are observed in the same order by all the members of a group. In addition, membership changes are totally ordered with respect to all regular messages. The model ensures that failures do not cause incomplete delivery of multicast messages. If two group members proceed from one view of membership to the next, they deliver the same set of messages

in the first view. The virtual synchrony model has been adopted by various group communication systems. Examples include Transis (Dolev and Malki, 1996), and Totem (Moser et al., 1996).

In the literature, there is a great deal of work on communication tools offering reliable multicast protocols for distributed applications (Birman, 1997). The Isis toolkit, developed at Cornell University, provided reliable multicast protocols supporting various ordered delivery properties such as causal and total ordering. It was one of the first available group communication systems providing multi-threading on top of Unix. It introduced the virtual synchrony model and has been used by several distributed applications including stock exchanges and air traffic control systems (Birman and van Renesse, 1994; Birman, 1993).

The Horus group communication system provides a flexible architecture where micro-protocols are composed to build high-level protocols depending on the needs of applications. Compared to its parent system Isis, it performs better and offers more flexibility for matching application requirements (van Renesse et al., 1994, 1996; van Renesse and Birman, 1995).

The Totem system offers reliable multicast communication guaranteeing totally ordered delivery on local area networks. It uses hardware broadcast property of such networks for achieving high performance. The system extends the virtual synchrony model, and is intended for distributed applications where fault-tolerance and real-time performance are critical (Moser at al., 1996).

The Transis system is a transport level reliable group communication service that distinguishes itself in allowing multiple network components to exist. It extends the virtual synchrony model for the purpose of supporting network partitions and consistent merging after recovery (Dolev and Malki, 1996; Malki, 1994). This approach to partitionable operation has been adopted by several systems including Horus and Totem.

Other example systems giving support for reliable multicast communication include Relacs (Babaoglu et al., 1995) and Rampart (Reiter, 1996).

The Ensemble system, developed as a successor project to the Horus, is a general-purpose group communication system providing the flexibility and performance required by several distributed applications. It also achieves a number of goals. Ensemble is a framework for conducting research in group communication protocols, and an implementation built in a functional programming language. It is designed to support the application of formal methods for the purpose of reasoning about the correctness of the protocols (Hayden, 1998).

Although protocols providing strong reliability guarantees are useful for many applications, they have some limitations. The drawback of protocols in this category is that in order to obtain strong reliability guarantees, costly protocols are used and the possibility of unstable or unpredictable performance under failure scenarios is accepted. These protocols allow limited scalability. As mentioned in (Piantoni and Stancescu, 1997) the maximum number of participants must not exceed about fifty to one hundred. Otherwise, transient performance problems can cause these protocols to exhibit degraded throughput.

2.2 Best-effort Reliability

This category includes scalable reliable multicast protocols that focus on best-effort reliability in large-scale systems. This class of protocols overcomes message loss and failures, but they do not guarantee end-to-end reliability. For instance, group members may not have a consistent knowledge of group membership, or a member may leave the group without informing the others. Example systems are Internet Muse protocol for network news distribution (Lidl et al., 1994), the Scalable Reliable Multicast (SRM) protocol (Floyd et al., 1997), the Pragmatic General Multicast (PGM) protocol (Speakman et al., 1998), the Xpress Transfer Protocol (XTP) (XTP Forum, 1995), and the Reliable Message Transfer Protocol (RMTP) (Paul et al., 1997; Lin and Paul, 1996).

SRM is a well-known reliable multicast protocol which was first developed to support wb, a distributed whiteboard application. The protocol is based on the principles of IP multicast group delivery, application level framing (ALF), adaptivity and robustness in the TCP/IP architecture design. Similar to TCP that adaptively sets timers or congestion

control windows, SRM algorithms dynamically adjust their control parameters based on the observed performance within a multicast session. It exploits a receiver-based reliability mechanism, and does not provide ordered delivery of messages. SRM protocol is designed according to the ALF principle that defers most of the transport level functionality to the application for the purpose of providing flexibility and efficiency in the use of the network. The protocol aims to scale well both to large networks and sessions.

PGM is a reliable multicast transport protocol that offers ordered, duplicate-free multicast data delivery. It guarantees that a receiver delivers all data packets or is able to detect unrecoverable data packet loss. PGM is designed with the goal of simplicity of operation for scalability and network efficiency. It employs a NAK-based error recovery mechanism and runs over a datagram multicast protocol such as IP multicast.

XTP is a general-purpose transport protocol designed to support a variety of applications ranging from real-time embedded systems to multimedia distribution over wide area networks. It provides all of the functionality found in TCP, UDP and TP4 plus new services such as multicast, multicast group management, transport layer priorities, rate and burst control, selectable error and flow control mechanisms, traffic descriptions for QoS negotiation.

RMTP is based on a hierarchical approach in which receivers are grouped into local regions. In each local region, there is a special receiver called a Designated Receiver (DR) which is responsible for processing ACKs from receivers in its region, sending ACKs to the sender and retransmitting lost packets. The sender only keeps information on DRs and each DR keeps membership information of its region. This approach reduces the amount of state information kept at the sender, end-to-end retransmission latency and the number of ACKs gathered by the sender. Since only the DRs send their ACKs to the sender, a single ACK is generated per local region and this prevents the ACK implosion problem.

This class of protocols is suitable for large-scale networks and they do scale beyond the limits of virtual synchrony protocols. When the message loss probability is very low or

uncommon, they can give a very high degree of reliability. But, failure scenarios such as router overload and system-wide noise which are known to be common in Internet protocols can cause these protocols to behave pathologically (Labovitz et al., 1997; Paxson, 1997).

2.3 Probabilistic Reliability

This thesis focuses on a new option in scalable reliable multicast protocols. We call this protocol bimodal multicast, or pbcast (probabilistic multicast) in short (Birman et al., 1999). This study demonstrates that bimodal multicast scales well, and in contrast to the other scalable reliable multicast protocols it gives predictable reliability even under highly perturbed conditions. The behavior of bimodal multicast can be predicted given simple information on how processes and the network behave most of the time. The protocol exhibits stable throughput under failure scenarios that are common on real large-scale networks. In contrast, this kind of behavior can cause other reliable multicast protocols to exhibit unstable throughput. Chapter 3 gives detailed information on bimodal multicast protocol.

2.4 Throughput Stability

For large-scale applications such as Internet media distribution, electronic stock exchange and distribution of flight telemetry data in air traffic control systems, the throughput stability guarantee is extremely important. This property entails the steady delivery of multicast data stream to correct destinations.

Traditional reliable multicast protocols depend on assumptions about response delay, failure detection and flow control mechanisms. Low-probability events caused by these mechanisms, such as random delay fluctuations in the form of scheduling or paging delays, emerge as an obstacle to scalability in reliable multicast protocols. For example, in a virtual synchrony reliability model, a less responsive member exposing such events can impact the throughput of the other healthy members in the group. The reason is as follows. For the reliability purposes, such a protocol requires the sender to buffer messages until all members acknowledge receipt. Since the perturbed member is less

responsive, the flow control mechanism begins to limit the transmission bandwidth of the sender. This in turn affects the overall performance and throughput of the multicast group. In effect, these protocols suffer from a kind of interference between reliability and flow control mechanisms. Moreover, as the system size is scaled up, the frequency of these events rises, and this situation can cause unstable throughput.

Throughput instability problem does not only apply to the traditional protocols using virtually synchronous reliability model. Scalable protocols based on best-effort reliability exhibit the same problem. As an example, recent studies (Liu, 1997; Lucas, 1998) have shown that, for the SRM protocol, random packet loss can trigger high rates of request and retransmission messages. In addition, this overhead grows with the size of the system. This thesis study includes a variety of results demonstrating the throughput instability problem in existing multicast protocols based on different reliability models.

2.5 Buffering

For error recovery, processes in a multicast session buffer the messages that they receive. Many reliable multicast protocols have all receivers buffer each message until it is guaranteed that the message has become stable, or has been delivered to every destination. In this case, the amount of buffering on each member is scaled up with group size. The reasons behind this buffering problem are as follows. As the group size is scaled up, the time to accomplish stability and to detect stability increases. In addition, depending on the application, the rate of sending multicast messages may grow.

Buffering scalability is an important issue for large-scale distributed applications that motivate our work. Very little attention has been paid to solve the buffer management problem in scalable reliable multicast protocols. Most existing protocols either ignore the problem, or provide only an ad hoc solution.

In general, work on buffering in group communication can be classified in three categories:

(a) *Multicast flow control* techniques attempt to control the amount of buffering using rate or credit-based mechanisms.

(b) *Stability optimization* techniques attempt to minimize the time to accomplish and detect stability of messages. This reduces the time that messages are buffered.

(c) *Memory reduction* techniques attempt to minimize the required amount of buffer memory.

Flow control techniques enable group members to manage their local buffers, and also deal with the problem of buffer overflow. A related work in this category is by (Mishra and Wu, 1998). They present two general-purpose flow control techniques, one conservative and one optimistic, and investigate the effect of these techniques on the performance of a group communication service. The conservative techniques prevent buffer overflow, but restrict the times when members can accept new multicasts. The optimistic techniques, on the other hand, are less restrictive. They minimize the possibility of buffer overflow, but do not prevent it completely. In the case of a buffer overflow, they offer mechanisms to tolerate overflow while ensuring correctness and progress of the multicast service. A simulation study is performed to compare these two flow control techniques in both ACK and NAK-based protocols. They conclude that an optimistic flow control technique is preferable to a conservative one most of the time.

In the second category, all reliable communication protocols try to optimize the time to achieve stability. The work in (Mishra and Kuntur, 1999) introduces a general technique called Newsmonger for improving the time to detect stability. The technique consists of a token rotating along a logical ring of group members, and is applicable to the atomic multicast protocols designed for asynchronous distributed systems. It is shown that it significantly improves the average stability time of multicast protocols. This approach is important when the application requires uniform or safe delivery of messages. As a beneficial side effect, it also reduces the amount of time that messages need to be buffered. The technique, when combined with our buffering optimization, is also useful to improve the latency of uniform delivery.

Another extensive study in this category focuses on buffer management mechanisms of reliable multicast protocols and investigates message stability detection protocols for large-scale reliable multicast communication (Guo, 1998). This study also introduces a gossip-style protocol with improved reliability and fault tolerance properties.

The buffer optimization techniques studied and evaluated in this thesis belong in the third category. The best known work in this category is a general protocol model called Application Level Framing (ALF) (Clark and Tennenhouse, 1990). ALF introduces the integration of the protocol levels from the transport level to the application level. This leaves many reliability decisions to the application. SRM is a well-known implementation of a multicast facility in the ALF model, and is used in various tele-conferencing applications. SRM does not buffer or order messages, and instead provides call-backs to the application when it detects message loss. The application decides whether and how to retransmit the message. Rather than buffering messages, the application may be able to regenerate messages based on its state.

2.6 Motivation and applications

Probabilistic protocols like pbcast provide weaker guarantees compared to other classes of multicast protocols with strong reliability guarantees. A probabilistically reliable multicast protocol is suitable for applications that are insensitive to small inconsistencies among participants. On the other hand, probabilistic communication protocols offer quality of service properties which are essential for some distributed applications. These properties are:

- Throughput stability guarantee which provides the steady delivery of multicast data stream to correct participants,
- Scalability of multicast communication as the number of participants increases,
- Minimal delivery latency of multicast messages.

One class of applications that can benefit from the properties provided by probabilistic protocols includes Internet media distribution applications that transmit media such as TV and radio, or teleconferencing data over the Internet. Such applications need to be

scalable, and they must tolerate some inconsistencies that may occur among the participants. For instance, it may be acceptable for a participant of an Internet TV application to miss some frames provided that the probability of such an event is very low. In addition, those applications disseminate media with a steady rate. An important requirement is the steady delivery of media by all correct participants in spite of possible failures in the system. Parameters of pbcast can be adjusted to meet those application needs.

Another application group is electronic stock exchange and trading environments like the Swiss Exchange Trading System (SWX) (Piantoni and Stancescu, 1997). In such systems, the trading information including orders and trades is multicast immediately to all members ensuring equal treatment and market transparency. A multicast communication protocol is used to disseminate trading information to all members at the same time and with minimal delay. Stock exchange and trading systems aim to serve large number of clients. SWX developers chose the Isis reliable group communication toolkit for this purpose, using it to implement fault tolerance with active replication. They observed some shortcomings that they attribute to the multicast protocols (and strong reliability guarantees) provided by Isis. For instance, one slow client could affect the entire system, especially under peak load. Also, multicast throughput was found to degrade linearly as the number of clients increased. This kind of shortcoming can be overcome using probabilistic protocols. In such systems, infrequent loss of a quote would not pose a problem as long as these events are rare enough and randomly distributed over messages generated within the system.

Air-traffic control systems require repeated refreshing of several types of data such as periodic updates to radar images and flight tracks. This kind of data changes rapidly, and infrequent dropping of updates would not cause a safety threat. Using a probabilistic protocol in this setting to transmit time-critical but less safety-critical data would guarantee stable throughput and minimal latency. Some data types in this kind of system may require stronger reliability guarantees, but such problems can be solved using virtually synchronous protocols “side-by-side” with the probabilistic ones. For example, France’s Phidias[1] air-traffic control system replicates flight plan updates within small

[1] http://www.stna.dgac.fr/projects/Phidias/

clusters of workstations using state machine replication. A flight plan is a record of the pilot's intentions and the instructions given by the controller. These updates need to be reliably multicast to the cluster participants.

In health-care systems, patient telemetry data are refreshed frequently on monitors located in places such as the patient's room, nursing station, and physician's office. Since infrequent loss of data of this sort is tolerable, they can be transmitted using probabilistic protocols. On the other hand, some data types, like medication change order, still need strong end-to-end guarantees. For example, a doctor making the dosage-changing operation at one end of the system needs the guarantee that the systems displaying medication order will reflect the changed dosage. Hence, for this data type, they require the use of traditional reliable multicast protocols with strong reliability guarantees.

The application classes described above are representative of a type of systems with mixed reliability requirements. They make use of two or more process groups. However, different uses of groups are independent. An application using a probabilistic protocol coexists with an application with stronger reliability needs. Traditional forms of reliable multicast should be used where individual data items have critical significance for the correctness and consistency of the application. Example data of this type include security keys for access to a stock exchange system, replicated flight plan data in air-traffic control centers, and medication dosage instructions in a health-care system. Other kinds of data match well to the probabilistic protocol's properties. Frequent message traffic such as periodic updates to radar images, refreshing patient telemetry can use probabilistic protocols safely.

2.7 Summary

This chapter provides background for reliable multicast protocols. Two classes of reliability guarantees, strong and best-effort, are described. Then, a new option in scalable reliable multicast protocols, probabilistic reliability is introduced. The throughput stability concept is explained, and buffering in the context of reliable multicast protocols is investigated. The chapter ends with the motivation, and application classes that this thesis study focuses on.

3. Bimodal Multicast Protocol

Bimodal multicast protocol is a new option in scalable reliable multicast protocols. The important aspects of bimodal multicast are an epidemic loss recovery mechanism, stable throughput property and a bimodal delivery guarantee. The protocol was first introduced by (Hayden and Birman, 1996) within the Ensemble system. This chapter gives information on the basis of epidemic communication, and describes bimodal multicast protocol suite that is the main focus of this thesis study.

3.1 Epidemic Communication

There exists a substantial class of large-scale distributed applications that are insensitive to small inconsistencies among participants, as long as these events are temporary and not frequent. Epidemic communication is suitable in this case where it allows such inconsistencies in shared data and offers low overhead as a benefit. Information changes are spread throughout the participants without incurring the latency and bursty communication that are typical for systems achieving a strong form of consistency (Golding and Taylor, 1992). In fact, this is especially important for large systems, where failure is common, communication latency is high and applications may contain hundreds or thousands of participants.

Epidemic communication mechanisms were first proposed for spreading updates in a replicated database. *Anti-entropy* is an epidemic communication strategy introduced for achieving and maintaining consistency among the sites of a widely replicated database. Compared to deterministic algorithms for replicated database consistency, this strategy

also reduces network traffic (Demers et al., 1987). Anti-entropy has been proposed as a mechanism that runs in background for recovering errors of direct mail in large network, as well (Birrell et al., 1982). Our protocol utilizes this mechanism for probabilistically reliable multicast communication. Periodically, every site chooses another site at random and exchanges information to see any differences and achieve consistency. This technique is called gossiping. For the case of database maintenance, the information exchanged during gossip rounds may include database contents. For epidemic multicast communication, the information may include some form of message history of the group members.

The anti-entropy method is based on the theory of epidemics (Bailey, 1975). According to the terminology of epidemiology, a site holding information or an update it is willing to share is called ‘infective’. A site is called ‘susceptible’ if it has not yet received an update. In the anti-entropy process, non-faulty sites are always either susceptible or infective. One of the fundamental results of epidemic theory shows that simple epidemics eventually infect the entire population. If there is a single infected process at the beginning, full infection is achieved in expected time proportional to the logarithm of the population size.

Epidemic or gossip style of communication has been used for several purposes. Examples include use of epidemic communication techniques for group membership tracking (Golding and Taylor, 1992), for support of replicated services (Ladin et al., 1992), for deciding when a message can be garbage collected (Guo, 1998) and for failure detection (van Renesse et al., 1998).

3.2 Prior Work

Bimodal Multicast protocol is inspired by prior work on epidemic protocols (Demers et al., 1987), Muse protocol for network news distribution (Lidl et al., 1994), the SRM protocol (Floyd et al., 1997), the NAK-only protocols of XTP (XTP Forum, 1995), and the lazy transactional replication method of (Ladin et al., 1992).

The work of (Demers et al., 1987) looked at systems under light load, and did not develop the idea of probabilistic reliability as a property one might present to the application developer. Moreover, since the frequency of database updates was very low, typically a few updates per second, this study did not consider the guarantee of stable throughput. Unlike the bimodal multicast model, they just assumed communication failures; bimodal multicast considers both process and communication failures.

The lazy replication technique of (Ladin et al., 1992) is based on the gossip approach. In this study, a replicated service consists of replicas running at different nodes in a network. The idea is executing an operation call at just one replica, while other replicas are updated by lazy exchange of gossip messages. The motivation is that for some distributed applications; a weaker causal operation order can preserve consistency while offering better performance. The technique is suitable for several applications such as distributed garbage collection and mail systems.

Bimodal multicast can also be considered as a soft real-time multicast protocol. Similar works are Δ-T protocol developed by (Cristian et al., 1985), and δ-causal protocol (Baldoni et al., 1996). These studies did not investigate the issue of steady load and steady data delivery during failures. They do not scale well. For instance, the Δ-T protocol involves delaying messages for a period of time proportional to the worst-case delay in the system, to estimates of the number of messages that might be lost and processes that might crash in a worst-case failure pattern. But, these delays would rise without limit as a function of system size. Similar concerns can be expressed about the δ-causal protocol, which guarantees causal order for messages while discarding the ones that are excessively delayed.

3.3 Inverted protocol stack

Traditional systems that suffer from throughput instability and scalability problems place reliability and ordering properties of protocols at bottom layers. One approach to overcome these problems is to construct large-scale reliable protocols using an inverted protocol stack. Probabilistic mechanisms are used at low layers, and reliability properties introduced closer to the application. Bimodal multicast protocol uses this inverted

protocol stack approach. The protocol is constructed using a novel gossip based transport layer. The transport layer employs random behavior to overcome scalability problems. Higher level mechanisms implementing stronger protocol properties such as message ordering and security can be layered over the gossip mechanisms. In this thesis, we focus on performance analysis of bimodal multicast and demonstrate how this approach works well on several network settings.

3.4 Properties of Pbcast Protocol

Bimodal multicast protocol, or pbcast for short, has the following properties:

Atomicity: The atomicity property of pbcast has a slightly different meaning than the traditional 'all-or-nothing' guarantee offered by reliable multicast protocols. Atomicity is in the form of 'almost all or almost none', which is called bimodal delivery guarantee. There is a high probability that each multicast will be delivered almost all participants, a low probability that a multicast will be delivered just a very small set of participants, and a vanishingly small probability that a multicast will be delivered by some intermediate number of processes.

Ordering: Each participant in the group delivers pbcast messages in FIFO order. In other words, multicast messages originated from a sender are delivered by each member in the order of generation at the sender. As mentioned in (Birman, 1997), stronger forms of ordering like total order can be provided by the protocol. (Hayden and Birman, 1996) includes a similar protocol providing total ordering.

Scalability: As the network and group size increase, overheads of the protocol remain almost constant or grow slowly compared to other reliable multicast protocols. This thesis study demonstrates that in both real and simulated network settings, most pbcast overheads are constant as a function of network and group size. In addition, throughput variation grows slowly with the log of the group size.

Throughput stability: Throughput variation observed at the participants of a group is low when compared to multicast rates. This leads to steady delivery of multicast messages at the correct processes.

Multicast stability detection: Pbcast protocol detects the stability of multicast messages. This means that the bimodal delivery guarantee has been achieved. If a message is detected as stable, it can be safely garbage collected. If needed, the application can be informed as well. Although some reliable multicast protocols like SRM do not provide stability detection, virtual synchrony protocols like the ones offered in Ensemble communication toolkit include stability detection mechanisms.

Loss detection: Because of process and link failures, there is a small probability that some multicast messages will not be delivered by some processes. The message loss is common at faulty processes. If such an event occurs, processes that do not receive a message are informed via an up-call.

3.4 Assumptions

For purposes of analysis, Pbcast assumes the following operating conditions (Birman et al., 1999):

- The protocol operates in a network for which throughput and reliability can be characterized for about 75% of messages sent, and where network errors *iid*.
- A correctly functioning process will respond to incoming messages within a known, bounded delay. This assumption needs to hold only for about 75% of processes in the network.
- Bounds on the delays of network links are known. However, this assumption is subtle, because Pbcast is normally configured to communicate preferentially over low-latency links.

3.5 Failure model

Process and communication failures in a distributed system can be classified into two broad types: *Hard* and *soft* failures. Hard failures include process crashes and network failures like partition events that persist long enough to trigger a timeout. Soft failures include events such as:

- Failure to receive a message that was correctly delivered. A buffer overflow can cause such a situation.
- Failure to respect time bounds for handling incoming messages.
- Transient network conditions that cause the network to locally violate its normal throughput and reliability properties.

Unlike reliable multicast protocols that only consider and tolerate hard failures, the goal of pbcast protocol is to overcome bounded number of soft failures as well. This is achieved with minimal impact on the throughput of multicasts sent by a correct process to other correct processes. Malicious (Byzantine) failures where a process or communication link can exhibit any behavior (e.g. sending or generating spurious and contradictory messages) are not considered in the Pbcast failure model.

3.6 Details of the protocol

Pbcast consists of two sub-protocols: an optimistic dissemination protocol and a two-phase anti-entropy protocol.

The former is a best-effort, hierarchical multicast used to efficiently deliver a multicast message to its destinations. This phase is unreliable and does not attempt to recover a possible message loss. If IP multicast is available in the underlying system, it can be used for this purpose. For instance, pbcast protocol implemented on top of ns-2 network simulator (Bajaj et al., 1999) in this thesis study uses IP multicast. Otherwise, a randomized dissemination protocol can play this role. For instance, the implementation of pbcast within Ensemble system (Hayden, 1998), which was ported to run on the SP2 parallel computer in this study, has used a hierarchical dissemination protocol.

The latter is an anti-entropy protocol that operates in a series of unsynchronized rounds. Each round is composed of two phases. The first phase is responsible for message loss detection. The second phase runs only if a message loss is detected, and corrects such losses.

3.6.1 Optimistic dissemination protocol

This stage of the protocol transmits each multicast message by means of an unreliable multicast primitive. Either IP multicast or a randomized dissemination protocol can be used for this purpose. For the randomized protocol, full connectivity of group members is assumed, and multicast spanning trees are superimposed upon the set of participants. Each process has pseudo-randomly generated spanning trees for disseminating messages to the whole group. Spanning trees are generated deterministically by using group membership information. A group member multicasts a message using a randomly selected spanning tree. A tree identifier is attached to the multicast message and the message is transmitted to the neighbors of the sender in the tree. When neighbors receive the message, they forward it using the same tree identifier. Pbcast implementation within the Ensemble system exploits a tree dissemination protocol for this first stage. The protocol uses Ensemble's group membership manager to track membership. But, this has the disadvantage of limited scalability, because Ensemble's group membership system can be scaled up to a few hundred members.

3.6.2 Two-phase anti-entropy protocol

This stage of the protocol is responsible for message loss recovery. It is based on an anti-entropy protocol that detects and corrects inconsistencies in a system by continuous gossiping. As mentioned before, the anti-entropy mechanism was previously used for error recovery in wide area database and large-scale direct mail systems (Demers et al., 1987; Birrell et al., 1982). The two-phase anti-entropy protocol progresses through unsynchronized rounds. In each round:

1. Every group member randomly selects another group member and sends a digest of its message history. This is called a ‘gossip message’.

2. The receiving group member compares the digest with its own message history. Then, if it is lacking a message, it requests the message from the gossiping process. This message is called ‘solicitation’, or retransmission request.

3. Upon receiving the solicitation, the gossiping process retransmits the requested message to the process sending this request.

Figure 3.1 illustrates the execution of pbcast protocol. A, B, C and D are group members, and the time advances from top to bottom. A dashed arrow in the figure denotes a message loss. First, multicast messages M0, M1 and M2 are transmitted unreliably by the dissemination protocol. Because of a process or communication failure, process C fails to receive message M0, and process D fails to receive M1. Then, the anti-entropy protocol executes. Each process selects another one randomly, and sends its message history digest. Upon receiving a gossip message from process B, process C discovers that it is missing M0 and requests a retransmission from B, and recovers this message loss. Because of the randomness in selecting a process to gossip, a process may not receive a gossip message in a given round. For example, process D does not detect its message loss until the next anti-entropy round. The figure simplifies the execution of pbcast by showing that the protocol alternates between dissemination and anti-entropy stages. But, in practice, these stages run concurrently.

One of the differences of pbcast’s anti-entropy protocol from the other gossip protocols is that during message loss recovery, it gives priority to the recent messages. If a process detects that it has lost some messages, it requests retransmissions in reverse order: most recent first. If a message becomes old enough, the protocol gives up and marks the message as lost. By using this mechanism, pbcast avoids failure scenarios where processes suffer transient failures and are unable to catch up with the rest of the system. One of the drawbacks of traditional gossip protocols is that such a failure scenario can slow down the system by causing processes’ message buffers to fill.

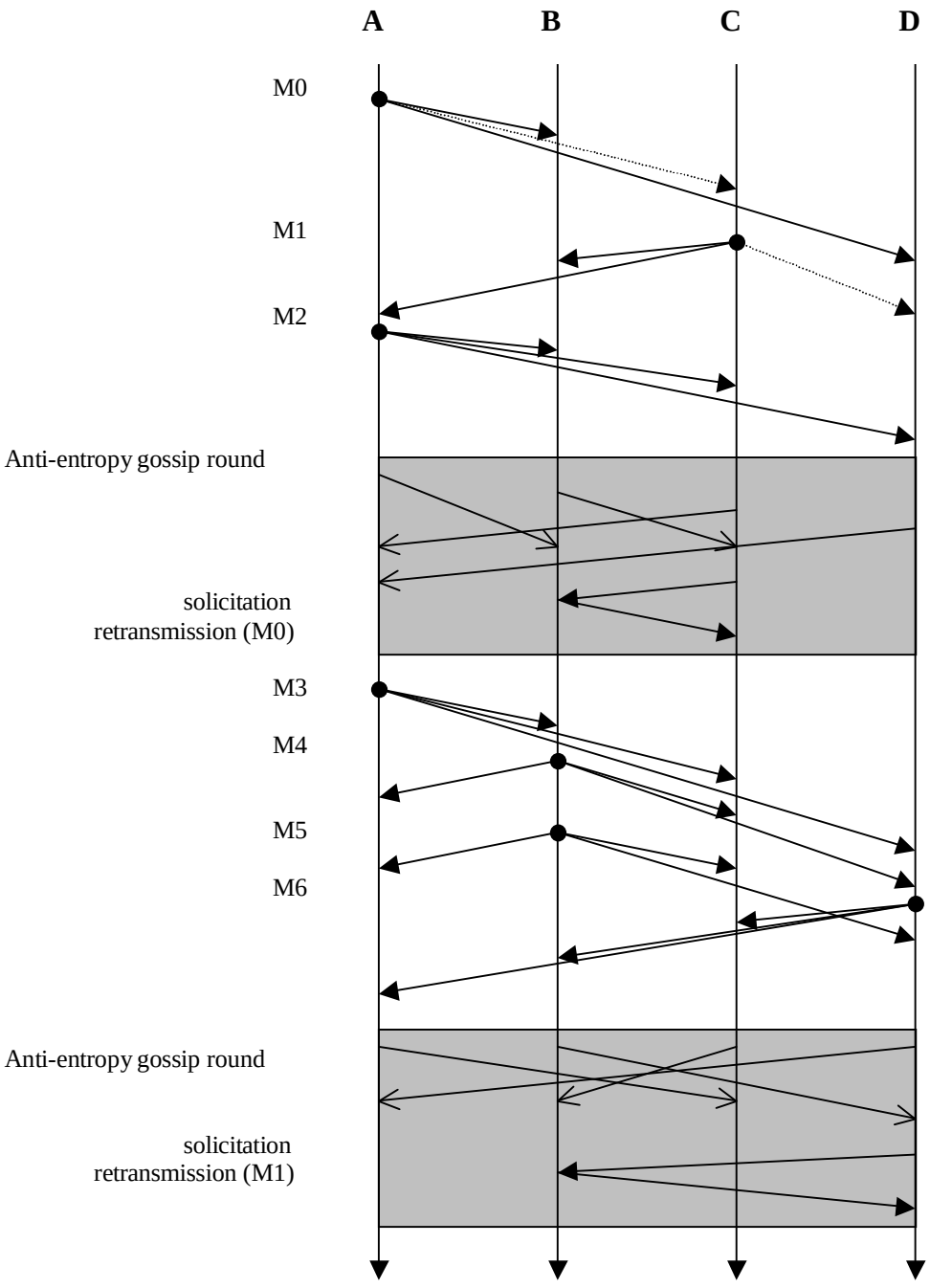


Figure 3.1 Execution of Pbcast Protocol

The duration of each round in the anti-entropy protocol is set to be larger than the typical round-trip time for an RPC over the communication links. The experiments and simulations conducted in this study use a round duration of 100msec.

Processes keep buffers for storing data messages that have been received from members of the group. Messages from each sender are delivered in FIFO order to the application. When a message is received, it is inserted in the appropriate location in receiver's message buffer. After a process receives a message, it continues to gossip about the message for a fixed number of rounds. Then, the message is garbage collected. The number of rounds during which the gossip continues for a given message and the number of processes to which a process gossips in each round are key parameters of the pbcast protocol. The product of these two parameters is called the 'fanout'. If a process can not recover a missing message, it is probable that other processes have garbage collected it. The process therefore marks a message as lost after a sufficiently long recovery period, and reports a gap to the application.

3.7 Optimizations to the anti-entropy protocol

When failure occurs, an anti-entropy protocol can enter a situation where failed processes affect correct processes by sending large number of retransmission requests. In order to limit such overheads, several optimizations are proposed for pbcast protocol. One of the contributions of this thesis is to investigate and analyze the effectiveness of these optimizations, using experimental and simulation methods. This section gives information on the optimizations we explored.

Soft failure detection

A retransmission message is sent in response to a solicitation message, if the solicitation message is received in the same gossip round for which the gossip message is sent. If a response takes longer than one round, this indicates the existence of a soft failure. The process or a link can be failed, and in this case a retransmission is likely to turn out to be a duplicate, because the same message will have been recovered elsewhere using healthy links. In such a situation, the retransmission message is not sent to the requesting process.

This optimization also avoids redundant retransmissions when a process, after recovering from a transient fault, finds many solicitations in its input buffers, and tries to respond to many solicitations at once.

Round retransmission limit

A process can limit its retransmissions to some maximum amount of data in one round. If more than this amount is requested, the process stops the retransmission when it reaches the limit. This optimization helps spreading the overhead both spatially and temporally. Retransmissions can be handled with different processes over several rounds.

Cyclic retransmissions

When a process responds to retransmission requests, it takes into account the messages it retransmitted in the previous rounds. If a message was retransmitted to the same destination in a previous round, or was retransmitted using IP multicast, it might still be in transit. Redundant retransmissions are avoided via this optimization.

Most-recent-first retransmission

If a process detects that it has missed more than one message, it requests retransmissions in reverse order: the most recent message is requested first. This optimization avoids scenarios in which a faulty process tries to catch up, but is unable to do so, and hence lags behind the rest of the group.

Independent numbering of rounds

Pbcast protocol progresses through asynchronous rounds. Each process manages its own round numbers. The round number is used for the decisions of garbage collection and message delivery. A gossip message also contains round number information. A process sending a solicitation message includes the round number sent by the gossiping process.

Optimizations that are described up to now are included in the basic pbcast protocol. The pseudo-code of the basic pbcast protocol is given in figure 3.2.

In this thesis study, we developed experimental and simulation models for Pbcast protocol. The experimental model uses basic Pbcast protocol implementation developed first (Hayden and Birman, 1996) and available in the Ensemble group communication system. The underlying network for our experimental model is the IBM SP2 parallel computer of the Cornell Theory Center. This work is described in chapter 4. In the simulation model, we designed and implemented basic Pbcast protocol and a number of optimizations on top of basic Pbcast. We used the ns-2 network simulator as the underlying structure. Chapter 5, 6 and 7 give details on the simulation model.

P: the set of processes in the system. N=|P|.

R: the number of rounds of gossip to run.

β: the probability that a process gossips to each other process. We define the *fanout* of the protocol to be β*N: this is the expected number of processes to which a participant gossips.

pbcast(msg):

```
add_to_msg_buffer(msg);
unreliably_multicast(msg);
```

first_reception(msg):

```
add_to_msg_buffer(msg);
deliver messages that are now in order;  report gaps after suitable delay;
```

add_to_msg_buffer(msg):

```
slot := free_slot;
msg_buffer[slot].msg := msg;
msg_buffer[slot].gossip_count := 0;
```

gossip_round:

```
my_round_number := my_round_number+1;
```

```
        gossip_msg := <my_round_number, digest(msg_buffer)>;
        for(i = 0; i < β*N/R; i := i+1)
                { dest := randomly_selected_member;     send gossip_msg to dest;   }
        for each slot
                msg_buffer[slot].gossip_count := msg_buffer[slot].gossip_count+1;
        discard messages for which gossip_count exceeds G, the garbage-collection limit;
rcv_gossip_msg(round_number, gmsg):
        compare with contents of local message buffer;
        foreach missing message, most recent first
                if this solicitation won't exceed limit on retransmissions per round
                        send solicit_retransmission(round_number, msg.id) to gmsg.sender;
rcv_solicit_retransmission(msg):
        if I am no longer in msg.round, or if have exceeded limits for this round
                ignore
        else
                send make_copy(msg.solicited_msgid) to msg.sender;
```

Figure 3.2 Pseudo-code for pbcast protocol

Multicast for some retransmissions

In the basic pbcast protocol, a process retransmits a message using unicast mode of communication. Some reliable multicast protocols like SRM employ multicast communication for retransmissions. But, this potentially increases the overhead of the protocol as the system size scales up. In this optimization to the pbcast protocol, each process keeps a counter to track the number of times a message is requested. If a message is requested twice, the process *multicasts* the corresponding retransmission to the entire group. The idea behind is that if at least two solicitations for the same message are

received, it is likely that this message loss affects more than one process in the group. Multicasting retransmission in this case may ease and speed up the recovery process. In this thesis study, we designed and implemented the optimization on ns-2 and analyzed its effectiveness. Information on the design and implementation is described in chapter 5, and results are given in chapter 6.

Gossip retransmit bit

When sending a gossip message, a process includes an additional bit for each message, which we call gossip retransmit bit, in its message digest. This bit indicates whether or not that message was retransmitted before. Based on this information, if a process is going to retransmit a message, it either uses multicast or unicast communication for retransmission. In this thesis study, we designed and implemented the optimization on ns-2 and analyzed its effectiveness. Information on the design and implementation is described in chapter 5, and results are given in chapter 6.

Local recovery

This optimization attempts to perform local error recovery. It utilizes neighborhood information among group members. If a group member determines that it lacks a message, then the member informs one of its neighbor members about the missing message. If the neighbor lacks the same message, this may be an indication of either a message loss affecting more than one group participant, or a local message loss affecting a sub-network. In this case, for achieving fast error recovery, the message source uses multicast for retransmission of the missing message. In this thesis study, we designed and implemented the optimization on ns-2 and analyzed its effectiveness. Information on the design and implementation is described in chapter 5, and results are given in chapter 6.

Efficient buffering

Buffering scalability is an important issue for large-scale distributed applications that motivate our work. In this thesis study, we implement some buffering optimization techniques on top of bimodal multicast, and demonstrate the efficiency of them by

extensive simulations. The most efficient optimization is based on the approach in (Ozkasap et al., 1999.b). Basically, the idea is to buffer each message on a small set of members, spreading the load of buffering over the entire membership. This causes each member to buffer not every message it received but only a small subset determined by a hash function. Results show that the optimization makes buffering scalable. In fact, the amount of buffering space per member actually decreases with group size. Information on this study is found in chapter 7.

Hierarchical gossip for scalability

The basic pbcast protocol seems to have two drawbacks in terms of scalability. First, each process needs a full membership information for the multicast group, since this information is required by the anti-entropy protocol. But, for large-scale groups, group membership information can become too large and membership updates cause high traffic on the network. Second, in a large network, anti-entropy protocol will involve communication over high-latency paths. Then buffering requirements and round length parameter of pbcast grow as a function of worst-case network latency.

It is possible to avoid these problems. A WAN is typically structured as a collection of LANs interconnected by TCP tunnels or gateways. In such an architecture, a hierarchical gossip approach would be suitable. Typical participants would only need to know about other processes within the same LAN component, only processes holding TCP endpoints would perform WAN gossip. In this case, only membership service needs the full membership information. A typical member would only know the members to which it gossips, and would gossip mostly to neighbors. Such an optimization also bounds the round length parameter of pbcast protocol, and in addition the protocol would have local costs.

3.8 Computational Model for Pbcast

A formal analysis of Pbcast protocol is given in (Birman et al., 1999). The analysis yields a computational model for Pbcast. In this section, we include the results for Pbcast's bimodal delivery distribution. The computational model assumes that the initial

unreliable multicast failed, that is only the sender initially has a copy of the message. The probability of message loss is 5% and the probability that a process will experience a crash failure during a run of the protocol is 0.1%. All of these assumptions are very conservative. Figure 3.3 illustrates Pbcast's bimodal delivery distribution for a range of group sizes. N denotes the group size. As shown in the figure, the protocol guarantees that the probability of almost none or almost all processes have delivered a multicast is high, and the intermediary outcomes are very low probability. When it is considered that the y axis is on a logarithmic scale, it becomes clear that Pbcast is likely to deliver to almost all processes if the sender remains healthy and connected to the network. The figure also shows that the risk of a failed Pbcast drops with the system size. For instance, the probability that only half of the processes in the group will receive a Pbcast, and the other half will fail to receive it, equals 10^{-16} for N=25, while the same probability equals 10^{-37} for N=100.

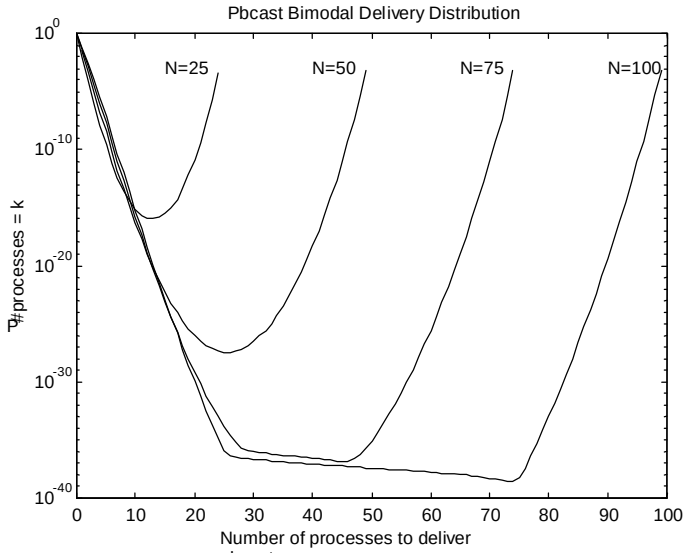


Figure 3.3. Pbcast's bimodal delivery distribution

3.9 Summary

This chapter starts by giving information on the epidemic communication. It then describes bimodal multicast protocol in detail. With this information, in the next chapters, we can start looking at models developed for the protocol, in this thesis study.

4. Experimental Model

We developed an experimental model for Pbcast protocol and virtually synchronous reliable multicast protocols of Ensemble group communication system. This chapter starts with discussing approaches to protocol performance evaluation. Then, we focus on our experimental study, analysis and comparison of results. The work uses an experimental model together with emulation methods for performance evaluation. We have designed experiments using the Ensemble toolkit on SP2 system, and constructed several group communication applications in order to investigate properties of Pbcast in practice. This study can be divided into the following categories:

a) Pbcast with soft process failures

b) Comparison with traditional and scalable Ensemble multicast protocol

c) Pbcast with system-wide message loss

4.1 Protocol Performance Evaluation

There are three primary approaches for evaluating protocol performance:

1. Analytical evaluation

2. Experimental model

3. Simulation model

Analytical evaluation demonstrates the impacts of operating parameters on a protocol's performance. Operating parameters such as number of group participants, link loss probability and message rate appear as variables in the formulas. However, analytical evaluation is only applicable and works for simplified models of the protocols. A formal analysis of the bimodal multicast protocol is found in (Hayden and Birman, 1996; Birman et al., 1999)

The second approach involves using a real implementation of a protocol to run experiments on real network settings. Doing so leads to realistic outcomes. But, this method may have some deficiencies in certain cases. For instance, it does not allow changing network load in a controlled way. Therefore, performance measurements express the protocol behavior in typical cases. But, performance evaluation under exceptional scenarios, such as networks with various failure models and link loss probabilities, is hardly possible. Emulation methods can be used to realize such network conditions to a certain extent. However, it is still not possible to control some operating parameters. For instance, network or group size and network topology is limited by the underlying real network's characteristics. That is why very little can be said about the protocol's behavior in very different scenarios. In addition, for a fair comparison of different protocols, identical network behavior is needed while running experiments repeatedly. This cannot be provided in a real network where background traffic changes dynamically due to other applications or processes running in the system. Experiments on an isolated network together with emulation capabilities modeling some network behavior would be suitable for comparison of different protocols.

The third approach is based on using simulation to construct a very detailed performance analysis of protocols. Simulation methods allow us gain power over all parts of the network and leads to better understanding of the protocol than the other approaches. For instance, in a simulation model, link loss probabilities can be set and maintained easily, several network topologies can be constructed. Many process group applications and scenarios can be built on top of these settings. Furthermore, to achieve protocol comparison it is possible to exchange one protocol for the other and rerun the simulator

under the identical network settings. However, simulations also have some drawback. For example, if a simulator does not model a protocol to be evaluated appropriately, results become unrealistic. Studies based on special-purpose simulators often do not reflect the richness of experience derived from experimentation with a more extensive set of traffic sources, queuing techniques and protocol models (Bajaj et al., 1999). General-purpose simulation tools prevent the disadvantages caused by special-purpose ones.

There are several network simulation tools available for protocol evaluation. Examples are REAL (Keshav, 1988) and ns-2 (Bajaj et al., 1999) network simulators.

This thesis study analyzes performance of bimodal multicast protocol and compares it with the other protocols using both experimental and simulation models. The experimental work has been performed on the SP2 system of Cornell Theory Center that offers an isolated network behavior. We used emulation methods to model process and link failures. Ensemble group communication system has been ported on SP2 and a detailed experimental study of pbcast protocol in Ensemble system and its comparison with Ensemble's virtual synchrony and scalable multicast protocols has been accomplished.

Our simulation study uses ns-2 network simulator to model network and protocol behavior. We have implemented pbcast and several optimizations to the protocol on ns-2. In contrast to the experimental study, simulation methods made possible to evaluate protocol's performance on several network topologies, failure models and large scale settings. Furthermore, we have been able to compare pbcast with a well-known scalable reliable multicast protocol SRM.

4.2 Experimental Platform

The RISC System/6000 Scalable Power Parallel System, or SP is a parallel computer from IBM. It consists of nodes connected by an ethernet and a switch. A node is a processor with associated memory and disk. Cornell Theory Center's SP2 system has total 160 nodes that fall into two types with the properties shown in figure 4.1. These nodes share data via message passing over a high performance two-level cross bar switch.

We ported Ensemble toolkit on this system, and designed many process group applications utilizing pbcast and Ensemble's traditional reliable multicast protocols.

	Thin nodes	**Wide nodes**
Amount	144	16
Speed (MHz)	120	135
Peak performance (MFLOPS)	480	540
Memory (Mbytes)	256	1024 or 2048

Figure 4.1. Node properties of SP2 system

4.3 Pbcast with soft process failures

In the first set of the experiments on SP2, our interest was in the performance of pbcast in the case of soft process failures. We emulate a process failure, such as a slow or overloaded member, by forcing the process to sleep with varied probabilities. We call a group member subject to such a failure as 'perturbed', and the probability of failure that impacts the process as 'perturb rate'. We have constructed process group applications on Ensemble toolkit for various group sizes starting from 8-member case up to 128-member process groups. There exists one sender process that disseminates 200 multicast messages per second to the group participants. During the execution of group application, some members were perturbed, that is forced to sleep during 100 millisecond intervals with varied perturb rates. First, we designed experiments so that one member is perturbed for various group sizes. Then, we increased the percentage of perturbed members up to 25% of the group size. In other words, we arranged the application so that, one or more group members would occasionally pause, allowing incoming buffers to fill and eventually overflow, but then resume computing and communication before the background failure detection used by the system have detected. This behavior is common in the real world, where multicast applications often share platforms with other applications.

An example application for an 8-member group, where one of the members is perturbed, is illustrated in figure 4.2.

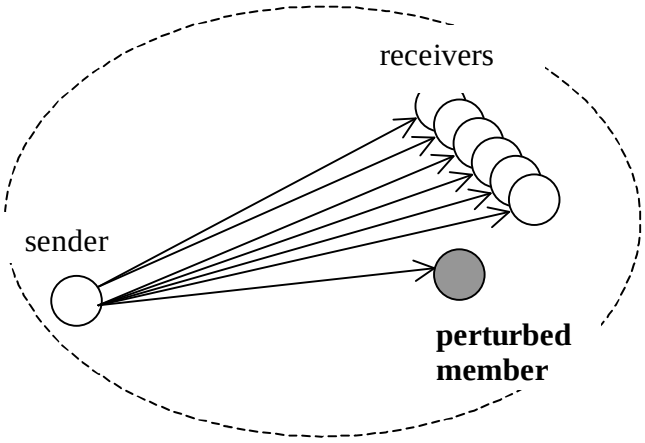


Figure 4.2. An 8-member process group with a perturbed process

4.3.1 Analysis and results

Based on the results of process group executions described above, we investigate the scalability and stability properties of pbcast. We mainly focus on the following analysis cases:

a) Throughput as a function of perturb rate for various group sizes

b) Throughput as a function of proportion of perturbed members

c) Protocol overhead associated with soft failure recovery as a function of group size

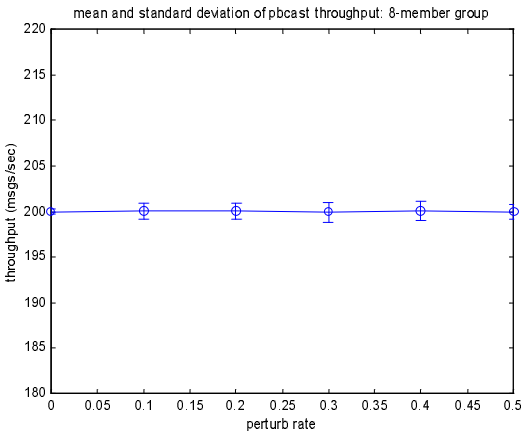
We varied a number of operating parameters. These are:

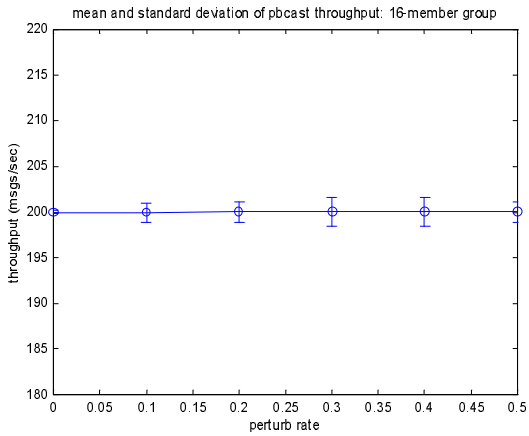
n: size of process group (8 to 128)

f: number of perturbed processes (1 to n/4)

p: degree of perturbation (0.1 to 0.9)

We measure throughput at the unperturbed or correct group members. The data points in the analysis correspond to values measured during 500 millisecond intervals. Since the throughput was steady, we also computed the variance of these data points. Figure 4.3 shows variation of throughput measured at a typical receiver as the perturb rate and group size increase. The group size is 8, 16, 96 and 128, respectively. These sample results are for the experiments where f=n/4. We can conclude that as we scale a process group, throughput can be maintained even if we perturb some group members. The throughput behavior remains stable as we scale the process group size even with high rates of failures. During these runs no message loss at all was observed at unperturbed members. On the other hand, the variance does grow as a function of group size. Figure 4.4 shows throughput variance as group size increases. Although the scale of our experiments was insufficient to test the log-growth predictions of computational results for pbcast (Birman et al., 1999), the data is consistent with those predictions. As we will see in the next section, the same conditions provoke degraded throughput for traditional virtually synchronous protocols.





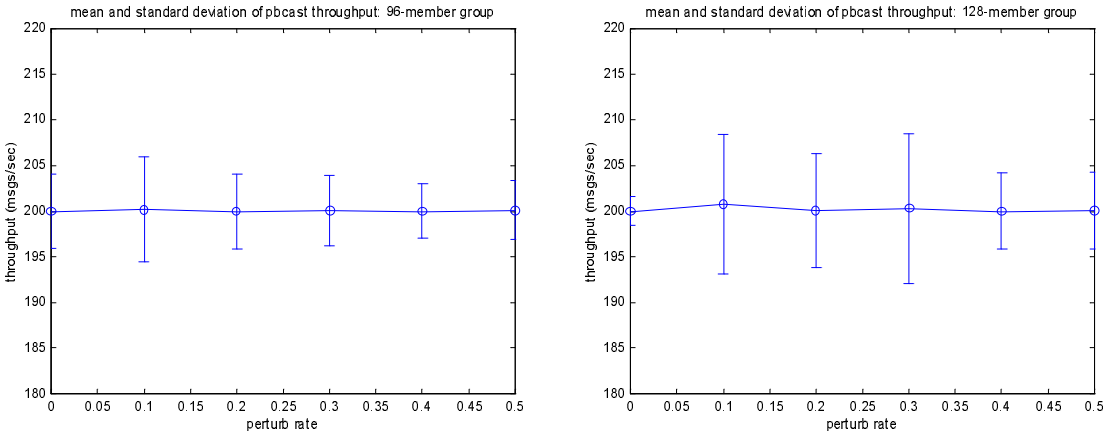


Figure 4.3. Variation of pbcast throughput

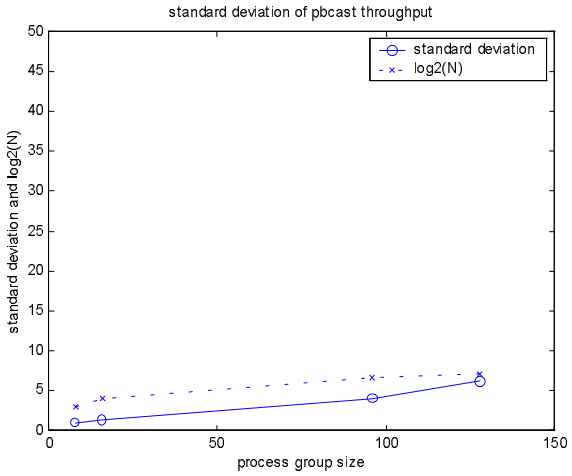


Figure 4.4. Throughput variance of pbcast as a function of group size

We analyzed protocol overhead associated with soft failure recovery, as well. For this purpose, retransmission behavior at a correct member was investigated. Figure 4.5 shows overhead as perturb rate increases, for 8, 16, 64 and 128-member groups, respectively. For these graphs f=n/4, and each region in the graphs illustrates data points measured during 500 msec intervals for a certain perturb rate. For instance, the first region contains data points for p=0.1, second one is for p=0.2, and so on. Figure 4.6.a superimposes the

data of four graphs in figure 4.5, and shows the percentage of messages retransmitted as p increases for various n.

For these experiments, we also compute the theoretical worst-case bounds for retransmission behavior at a correct member (figure 4.6.b). Assume r is the number of multicast data messages per second disseminated to the group by the sender, and p is the perturb rate. In every 100 msec (which is the duration of a gossip round in the experiments), at most ((r/10)*p) messages are missed by a faulty member, and a correct member gossips to two randomly selected group members. In the worst-case, if these two members are faulty and they lack all ((r/10)*p) data messages, they request retransmissions of these messages from the correct member. Then, the correct member retransmits at most 2*((r/10)*p) = (r*p)/5 messages in every 100 msec. In our experiments, we measured data points during 500 msec intervals, and computed the percentage of retransmitted messages to the multicast data messages disseminated by the sender during each interval. If we compute theoretical values for 500 msec intervals, the correct member retransmits at most 5*(r*p)/5 = r*p messages, and the sender disseminates r/2 messages during every 500 msec interval. Then, the bound for the percentage of retransmitted messages would be (r*p)/(r/2) = 2*p in the worst-case. Figure 4.6.b shows the computed theoretical worst-case bounds. Note that, our experimental results are below the theoretical bound, and the results confirm that overhead on the correct processes is bounded as the size of process group increases. But, in our experiments, as the group size increases, we observed an increase in the percentage of retransmitted messages. We believe, this is mainly due to the increase in the number of perturbed members with the group size. Because, in these experiments, number of perturbed members equals 25% of the group size (f = n/4).

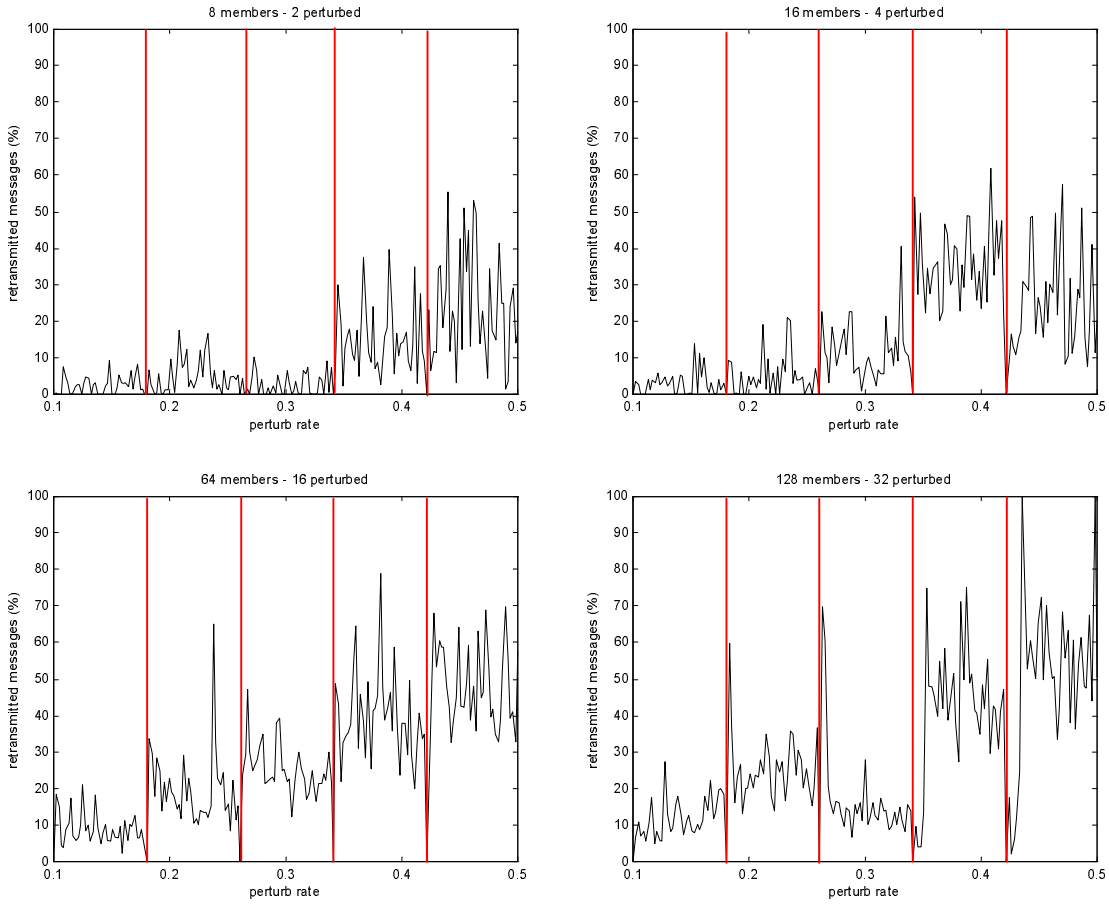
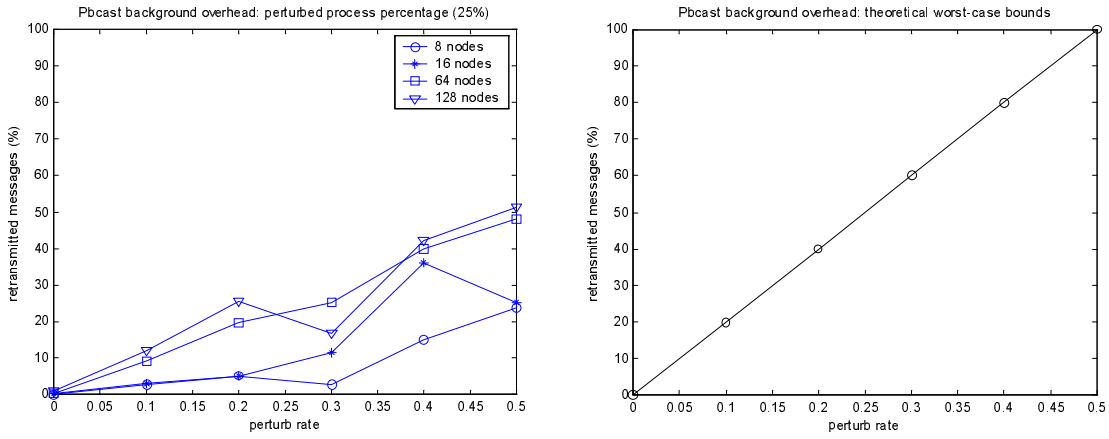


Figure 4.5. Pbcast overhead associated with soft failure recovery



(a) (b)

Figure 4.6. Percentage of message retransmissions as a function of p. a) Experimental results, b) Theoretical worst-case bounds

4.4 Comparison with traditional and scalable Ensemble multicast protocol

In this section, we focus on the throughput behavior of Ensemble's traditional and scalable multicast protocols, and compare them with pbcast. We used the same experimental settings described in the previous section. The group application utilized Ensemble's multicast protocols based on the virtual synchrony reliability model. One of the group members is a sender that disseminates 200 multicast messages per second. Message size is 7Kbytes. Up to 25% of receiver processes are perturbed.

4.4.1 Analysis and results

Based on the results of process group executions, we investigate and analyze the throughput behavior of two protocols. We varied operating parameters n, f and p. We measure throughput at the unperturbed or correct group members. The data points in the analysis correspond to values measured during 500 millisecond intervals. Figure 4.7 shows some analysis results for 32, 64 and 96-member process groups. Graphs show the superimposed data for cases f=1 and f=n/4. We see that even a single perturbed group member impacts the throughput of unperturbed members negatively. The problem

becomes worse as the group size (n), percentage of perturbed members (f), and perturb rate (p) grow. If we focus on the data points for a single perturb rate, we see that the number of perturbed members affects the throughput degradation. For instance, in figure 4.7, for a 96-member group when the perturb rate is 0.1, the throughput on non-perturbed members for the scalable Ensemble multicast protocol is about 90 messages/second when there is one perturbed member in the group. The throughput for the same protocol decreases to about 50 messages/second when the number of perturbed members is increased to 24. The same observation is valid for the traditional Ensemble multicast protocol. Among the two protocols, the traditional Ensemble multicast protocol shows the worst throughput behavior. Figure 4.8 shows the impact of an increase of group size on the throughput behavior clearly, when f=1. In the previous section, we showed that, under the same conditions, pbcast achieves the ideal output rate even with high percentage of perturbed members.

We can conclude that pbcast is more stable and scalable compared to the traditional multicast protocols. The fragility of the traditional multicast protocols becomes evident very quickly, once the perturbed process begins to sleep for long enough to significantly impact Ensemble's flow control and windowed acknowledgement. Furthermore, in such a condition, high data dissemination rates can quickly fill up message buffers of receivers, and hence can cause message losses due to buffer overflows.

In the case of virtually synchronous protocols, a perturbed process is particularly difficult to manage. Since the process is sending and receiving messages, it is not considered to have failed. But, it is slow and may experience high message loss rates, especially in the case of buffer overflows. The sender and correct receivers keep copies of unacknowledged messages until all members deliver them. It causes available buffer spaces to fill up quickly, and activates background flow control mechanisms. Setting failure detection parameters more aggressively has been proposed as a solution (Piantoni and Stancescu, 1997). But, doing so increases the risk of erroneous failure detection approximately as the square of the group size in the worst-case. Because, all group members monitor one another and every member can mistakenly classify all the other (n-1) members as faulty where n is the group size. Then, the whole group has n*(n-1) chances to make a mistake during failure detection. Since the failure detection parameters

are set aggressively in such an approach, it is more likely that randomized events such as paging and scheduling delays will be interpreted as a member's crash. As group size increases, failure detection accuracy becomes a significant problem. Most success scenarios with virtual synchrony use fairly small groups, sometimes structured hierarchically. In addition, the largest systems have performance demands that are typically limited to short bursts of multicast.

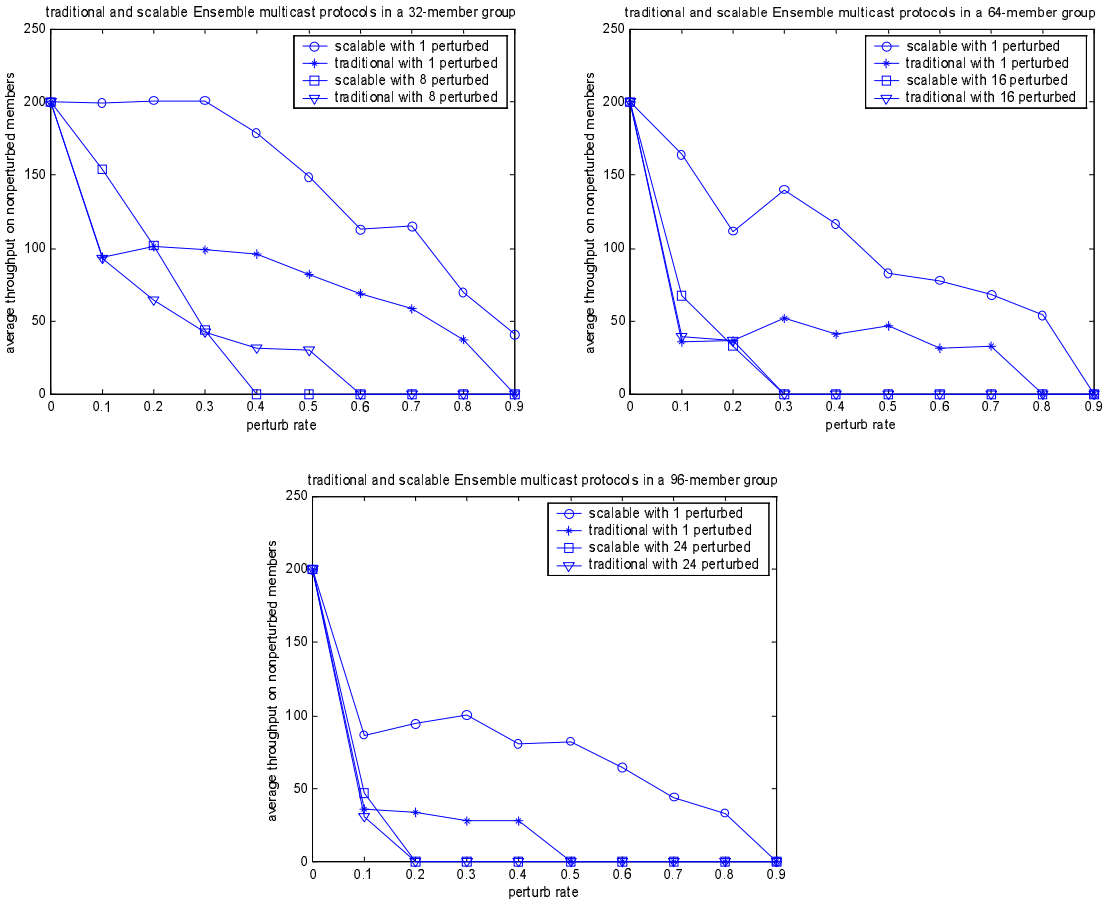


Figure 4.7. Throughput performance of Ensemble's reliable multicast protocols

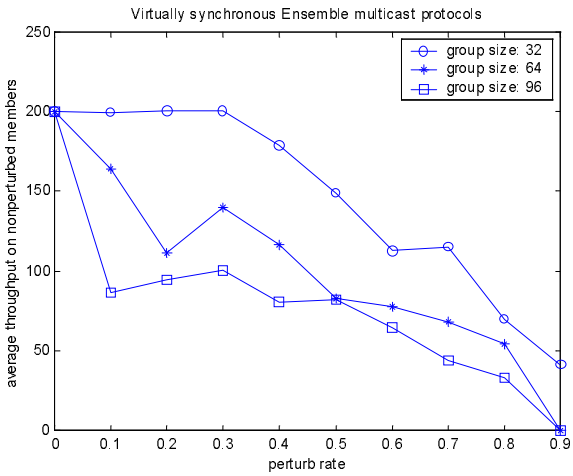


Figure 4.8. Throughput behavior as a function of group size

4.5 Pbcast with system-wide message loss

In this section, our interest lies in the behavior of pbcast while link noise occurs among members of the process group. One feature of pbcast is that, in the case of high loss and data rates, the protocol is capable of reporting message losses to correct processes. We emulate link failures or network load by randomly dropping messages with a given probability. We call the probability of a message loss between two participants the ‘drop rate’. When we apply a given drop rate among all participants, this defines the ‘system-wide drop rate’. We have constructed process group applications for various group sizes. One of the group members is the sender that disseminates multicast data at a given rate. We apply various system-wide noise rates to the network.

4.5.1 Analysis and results

Based on the results of several process group executions, basically we focus on the analysis of the impact of message loss on pbcast reliability as a function of group size, message drop rate, and multicast data rate.

We varied the following operating parameters:

n: size of process group (8 to 128)

r: multicast data rate (low bandwidth: 100 messages per second, high bandwidth: 200 messages per second)

d: system-wide drop rate (0.02 to 0.2)

We measure throughput at receiver processes. The data points in the analysis correspond to values measured during 500 millisecond intervals. Figure 4.9 gives analysis results showing the impact of message loss on pbcast reliability. At low multicast dissemination rates, we find that even significant system-wide noise can be tolerated. No message loss at all observed for this case. On the other hand, at high multicast data rates, noise triggers message loss in large groups. The reason can be explained as follows. At high multicast data rates, system-wide drop rate can cause the loss of higher number of data messages compared to the case for low multicast data rates. This situation triggers higher control message traffic for loss recovery. Since the system-wide drop rate affects control messages as well as data messages, this can lead to failures during loss recovery and hence can cause message loss. In this case, pbcast reports gaps in multicast data stream to the members. We observe that, with a mixture of high data bandwidths and high drop rates, Pbcast is quite capable of reporting gaps to correct processes. This is a feature of the protocol. In the same situation, a virtual synchrony protocol would refuse to accept new multicasts. As discussed in the previous section, such a scenario would cause a degraded performance for virtually synchronous multicast protocols.

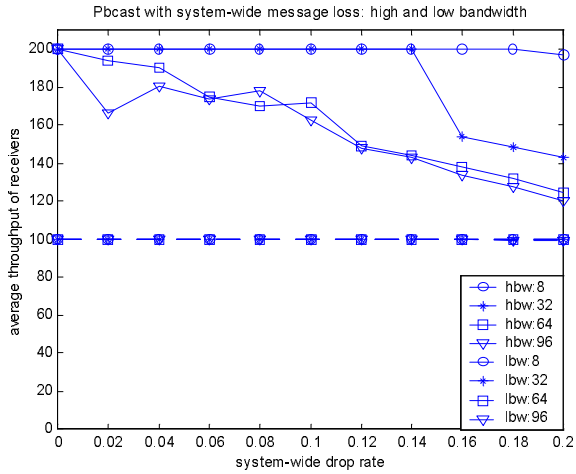


Figure 4.9. Impact of message loss on pbcast reliability

4.6 Discussion

Our experimental study yields some general conclusion about the behavior of basic Pbcast and virtually synchronous multicast protocols. In the first part of the study, we focused on the performance of Pbcast in the case of soft process failures. We showed that the throughput behavior of Pbcast remains stable as we scale the process group size even with high rates of failures. Furthermore, our results confirm that overhead on the correct processes is bounded as the size of process group increases. Then, we compared basic Pbcast with virtually synchronous Ensemble multicast protocols in the case of soft process failures. We showed that even a single perturbed group member impacts the throughput of unperturbed members negatively. On the other hand, Pbcast achieves the ideal throughput rate even with high percentage of perturbed members. We concluded that Pbcast is more stable and scalable compared to the traditional multicast protocols. Finally, we analyzed the impact of system-wide message loss on Pbcast reliability. We showed that, at low multicast dissemination rates, even significant system-wide noise can be tolerated. On the other hand, at high multicast data rates, noise triggers message loss in large groups. In this case, Pbcast reports gaps in multicast data stream to the members.

4.7 Summary

This chapter first studies three primary approaches for protocol performance evaluation; analytical evaluation, experimental model and simulation model, along with the advantages and disadvantages of each approach. We then describe the results and analysis of the experimental model developed in this thesis study. The analysis is studied in three categories: Pbcast with soft process failures, comparison with traditional and scalable Ensemble protocols, and Pbcast with system-wide message loss. The chapter ends with a discussion on the general results of our experimental study.

5. Simulation Model for Basic Pbcast, optimizations and SRM

In this thesis study, a simulation model for basic Pbcast and some optimizations to the protocol are designed and implemented. This chapter focuses on the design and implementation of our model. We use ns-2 network simulator as the underlying environment. Ns-2 provides support for SRM (Scalable Reliable Multicast) protocol as well. By using our simulation model, we investigated and analyzed the behavior of Pbcast and SRM protocols on various network conditions. This chapter also gives information on the simulator and the SRM protocol.

5.1 Simulator

Simulation in network research has a significant role of providing an environment in which to develop and test new network technologies without the high cost and complexity of constructing test-beds (Bajaj et al., 1999). Simulation allows the evaluation of network protocols under various network conditions and scenarios. Investigating protocols and their interaction with other protocols, and comparing them with other approaches under a wide range of conditions is critical to explore and understand the behavior and characteristics of protocols.

In this study, we choose ns-2 as the simulation environment. Ns-2 (Bajaj et al., 1999; Fall and Varadhan, 1999) is a discrete event simulator for networking research. It began as a variant of the REAL network simulator and is used widely by many network researchers. It is an object-oriented simulator implemented in C++, and uses OTcl as the command

and configuration interface. Basic elements of the simulator are nodes, links and agents. Agents represent endpoints where network-layer packets are constructed and consumed.

Ns-2 provides support for various networking concepts such as routing, multicast protocols (e.g. IP multicast and SRM), link error models, topology and traffic generation. In addition, it supports development and evaluation of new protocols, repetition of simulations under controlled conditions which makes it especially convenient for comparing several protocols under the same network settings. Also, support for simulation needs such as abstraction, emulation, scenario generation, visualization and extensibility is provided.

5.2 Basic Pbcast Design and Implementation

We have discussed Pbcast protocol in detail in Chapter 3. Our basic Pbcast protocol design on ns-2 consists of three modules as shown in the block diagram of figure 5.1. The bottom module that performs unreliable data dissemination uses IP multicast protocol. The second module is the gossip based anti-entropy protocol. The third module accomplishes FIFO message ordering. Total number of lines for the code of the implementation is approximately 1500.

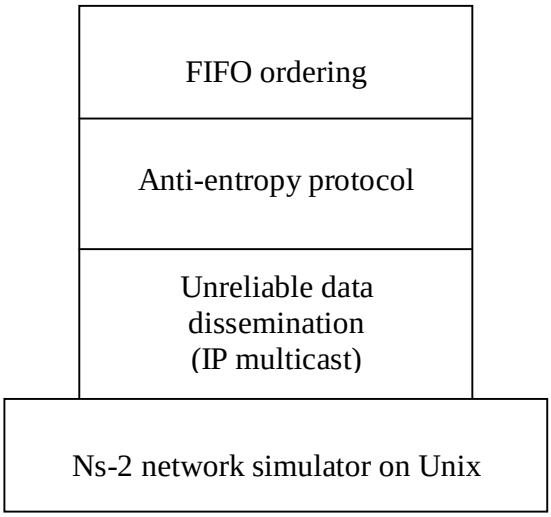


Figure 5.1. Basic pbcast design on ns-2

Our design follows an event-based approach. We define four message types for data, gossip, request and retransmission messages. Each Pbcast message has a type field. In addition, messages contain the following fields, where the first entry is the type of the corresponding message.

Data message: <PT_PBCAST_DATA, sequence number, data>

Gossip message: <PT_PBCAST_GOSSIP, message buffer, round number>

Request message: <PT_PBCAST_REQUEST, sequence number of the requested message, round number>

Retransmission message: <PT_PBCAST_RETRANS, sequence number of the message retransmitted, data>

Every member has a message buffer for keeping data messages received, for some predefined number of rounds (called stability threshold) after which they are garbage collected. A message buffer entry for a data message consists of the message content and gossip count of the message. The gossip count of a message is initially 0, and incremented at each gossip round.

Basic pbcast protocol agent has the following operating parameters:

sub-gsize: number of members to gossip in each round.

step-interval: gossip round duration. Default is 100msec.

limit-retrans: maximum number of messages that can be retransmitted by a member in one round.

limit-requests: maximum number of request messages that can be sent by a member in one round.

stable-threshold: stability threshold value for garbage collection. Default value is 10.

We define the following four events that trigger the protocol actions:

1. Receipt of Pbcast data or retransmission message

2. Receipt of Pbcast request message

3. Receipt of Pbcast gossip message

4. Timer interrupt for gossip round

When a member receives a data or a retransmission message, it updates the buffer and deliver messages that are now in order. Also, if some messages are declared as lost, the application is informed about them.

When a member gets a request message, the member checks its round number and retransmission count. If it is still in the same round with the round number in the request message, and it has not exceed retransmission limits for current round, then it retransmits the requested data message to the requestor.

When a member receives a gossip message, it first compares its message buffer with the message buffer digest in the gossip message. For each missing message, with the most recent one first, if the member has not exceeded request limits for current round, then it sends a request message to the sender of gossip message.

When the timer for current gossip round of a member expires, the member increments its round number, resets its request and retransmission counters. Then, it sends its gossip message to randomly selected members defined by *sub-gsize* parameter, and schedules the timer to *step-interval* value for the next gossip round.

Algorithm for basic Pbcast agent of our simulation model is given in figure 5.2. The Pbcast agent runs at every member of a process group application communicating via Pbcast protocol. In the algorithm, msg denotes a message received by a member. Figure 5.3 gives algorithms for the functions update_msg_buffer, deliver_if_in_order, send_subg and garbage_collect_stable_msgs used by basic Pbcast agent.

```
Switch (event)
{
        Case: Receipt of PT_PBCAST_DATA or PT_PBCAST_RETRANS
        {
        update_msg_buffer(msg.seqno)
        deliver_if_in_order()
        }
        Case: Receipt of PT_PBCAST_REQUEST
        {
        if ((my_round_number == msg.round_number) and (retrans_count_ < limit-retrans)) {
                send_retrans(msg.source, msg.seqno, data message)
                retrans_count ++ }
        }
        Case: Receipt of PT_PBCAST_GOSSIP
        {
        compare my_msg_buffer with msg.msg_buffer
        for each missing message with msg_id { // most recent message first
                request_count ++
                if (request_count < limit-requests )
                        send_req(msg.source, msg.round_number, msg_id)
                else break
                        }
        }
        Case: Timer interrupt for gossip round
        {
        my_round_number_ ++
        reset request_count and retrans_count
        send_subg(PT_PBCAST_GOSSIP)
        schedule_timer(step-interval)
        }
}
```

Figure 5.2. Algorithm for basic pbcast protocol on ns-2

```
update_msg_buffer(seqno) {
        Put message with seqno in my_msg_buffer
        Set its gossip_count to 0
}
deliver_if_in_order() {
        deliver messages in order to the application
        inform application about LOST messages
}
send_subg(msg_type) {
        get_subgroup(sub_group, sub-gsize)
        for each member_ in the sub_group {
                allocate a packet msg
                msg.round_number = my_round_number
                msg.msg_buffer = my_msg_buffer
                msg.type=msg_type
                send(member_, msg)
                deliver_if_in_order()
                garbage_collect_stable_msgs()
        }
        increment gossip_count of each msg in the message buffer
}
garbage_collect_stable_msgs() {
        for messages in my_msg_buffer
                garbage collect a message if its gossip_count > stable-threshold
                declare a message old enough as lost
}
```

Figure 5.3. Algorithms for some functions of pbcast

5.3 Optimizations to Basic Pbcast

Based on the results of analysis studies, in order to improve latency characteristics and reliability properties of basic pbcast protocol, we propose some optimizations to the protocol. In this section, we describe design and implementation of the optimizations. We

also propose some techniques for efficient buffering of reliable multicast protocols. They are discussed separately in chapter 7.

5.3.1 Pbcast-ipmc protocol

The basic pbcast protocol uses point-to-point communication when retransmitting a message. If a message is requested more than once, it is likely that this message loss affects not just one member of the group. Thus, it is more appropriate to use multicast communication for retransmission in such a scenario. We model this optimization that we call pbcast-ipmc on ns-2, and show that it leads to fast error recovery and better reliability than basic pbcast under the same network conditions.

A request counter for every message in the message buffer is needed as an additional data structure. Initially, request counter for a message is set to 0. Figure 5.4 gives the modifications needed for pbcast-ipmc protocol. If a member receives a retransmission request for a message in its buffer, then its request counter is incremented. When sending a retransmission for message, its request counter is checked. If it exceeds a certain threshold, then instead of unicast, the member does *multicast* the retransmission message to the group. In our implementation, we set the threshold to two.

At first glance, pbcast-ipmc has the following advantages. It decreases the request message traffic compared to basic pbcast especially when message losses affect more than one member in the group. Since the optimization exploits IP multicast during loss reovery, it increases reliability of the protocol where there exists random noise on the links. On the other hand, retransmission message traffic is expected to increase in certain conditions due to the use of multicast communication. However, our analysis study shows that overall bandwidth requirement of pbcast-ipmc is in fact the same as basic pbcast.

Initially, request_counter for every message is set to 0.

Case: Receipt of PT_PBCAST_REQUEST

{

msg.request_counter ++

```
// multicast retransmission to group
if (msg.request_counter >= threshold) {
        send_retrans(groupid, msg.seqno, data message)
        retrans_count ++
        msg.request_counter = 0 }
else {
        // basic pbcast: unicast retransmission
        if ((my_round_number == msg.round_number) and (retrans_count_ < limit-retrans))
                send_retrans(msg.source, msg.seqno, data message)
                retrans_count ++ }
}
```

Figure 5.4. Algorithm for pbcast-ipmc protocol

We will now describe a simple scenario to show how pbcast-ipmc performs better than basic pbcast in certain network conditions. We assume that there is a 5-member process group with members A, B, C, D and E. Member A is the sender and it multicasts data messages to the group. It first multicasts message M1, but assume that only B receives M1, and due to some temporary link failure or noise, members C, D and E fail to receive M1. Then, the sender continues multicasting data messages to the group. All members successfully receive successive multicast data messages. During gossip rounds, each member randomly chooses another member and conveys its gossip message. The parameter *sub-gsize* equals 1. In the first round, assume that member A gossips to member C, and similarly B to A, C to D, D to B, and E to C. In the second gossip round, assume that A, B, C, D and E gossip to E, C, A, A, and D respectively. Under these assumptions, figure 5.5 presents the execution of basic Pbcast protocol. Likewise, figure 5.6 illustrates the run of pbcast-ipmc under the same conditions. Protocol executions proceed as follows:

On receiving a gossip message from process A, process C finds out that it lacks data message M1. It then sends a request for M1 to process A. Until now, both Pbcast and pbcast-ipmc do the same actions. For pbcast-ipmc, A increments request counter for M1 on receiving the request from C. We assume threshold equals 0. Since request counter value M1 is not greater than or equal to threshold value, process A responds this request

by retransmitting M1 to C in unicast mode. Similarly, for basic pbcast, process A retransmits M1 to C. After the first gossip round, for both protocols C recovers message loss, and it is able to receive M1. In the second gossip round, on receiving a gossip message from A, process E realizes that it lacks M1, then it immediately requests M1 from A. For pbcast-ipmc, A increments request counter for M1 again. Now, request counter of M1 equals threshold, and A *multicasts* retransmission M1 to the group. As a benefit of pbcast-ipmc optimization, process E now recovers message loss, and process D does so, as well. Thus, all members deliver M1 at this point. For basic pbcast, on the other hand, A retransmits M1 to E in unicast mode. Then, process E receives M1. After the second gossip round, process D still lacks M1, it would be able to recover the loss in successive rounds of gossip. These sample runs of basic pbcast and pbcast-ipmc illustrate that pbcast-ipmc increases probability of rapid convergence during loss recovery.

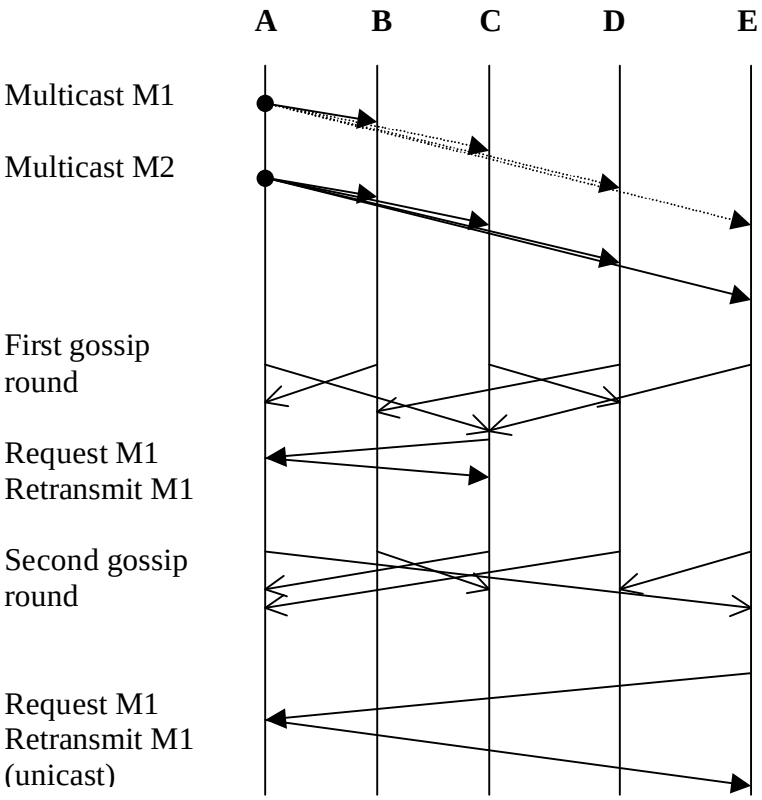


Figure 5.5. A sample run of basic pbcast protocol

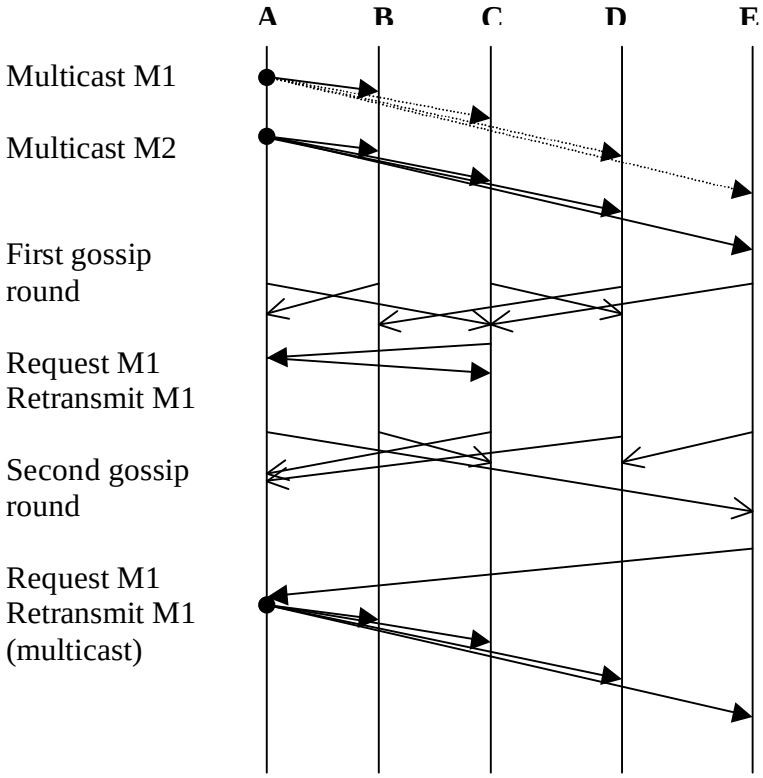


Figure 5.6. A sample run of pbcast-ipmc protocol

5.3.2 Pbcast-grb protocol

grb stands for gossip retransmit bit. This optimization consists of pbcast-ipmc together with the idea of gossip retransmit bit. It keeps information about whether or not a message is retransmitted before. Based on this information, members either use multicast or unicast for retransmission. We model pbcast-grb on ns-2, and show that, similar to pbcast-ipmc it leads to fast error recovery and better reliability than basic pbcast under the same network conditions. We observe that pbcast-grb has almost the same behavior as pbcast-ipmc in terms of loss recovery and reliability.

For this optimization, members keep a retransmit for every message in their buffer. If a member retransmits a message, it sets the retransmit bit of that message. When sending a gossip message, members include this information, that we call gossip retransmit bit, for

each message in their message digest. When a member receives a gossip message, it makes necessary updates on the retransmit bits of messages in its own buffer. If a member is going to retransmit a message and the retransmit bit of that message is set, then it sends retransmission in unicast mode. Otherwise, it sends retransmission by using multicast mode. After retransmission is performed, it sets the retransmission bit of the message. Figure 5.7 gives the modifications needed for pbcast-grb protocol.

Similar to pbcast-ipmc, pbcast-grb has advantages over basic pbcast protocol in terms of fast and easy error recovery. It utilizes multicast for some retransmissions based on the retransmit bit information.

Initially, my_retransmit_bit and gossip_retransmit_bit for every message is set to 0.

When sending a retransmission message, increment my_retransmit_bit and gossip_retransmit_bit of that message

Case: Receipt of PT_PBCAST_REQUEST

```
{
msg.request_counter ++
// multicast retransmission to group
if ((msg.request_counter >= threshold) or
((msg.my_retransmit_bit == 1 ) and (msg.gossip_retransmit_bit >= 1))) {
        send_retrans(groupid, msg.seqno, data message)
        retrans_count ++
        msg.request_counter = 0
        msg.my_retransmit_bit = 0
        msg.gossip_retransmit_bit = 0 }
else {
        // basic pbcast: unicast retransmission
        if ((my_round_number == msg.round_number) and (retrans_count_ < limit-retrans)) {
                send_retrans(msg.source, msg.seqno, data message)
                retrans_count ++
                msg.my_retransmit_bit = 1 }
        }
```

```
}
Case: Receipt of PT_PBCAST_GOSSIP
{
compare my_msg_buffer with msg.msg_buffer
for messages with msg.gossip_retransmit_bit >= 1
        increment their local gossip_retransmit_bit
for each missing message with msg_id { // most recent message first
        request_count ++
        if (request_count < limit-requests )
                send_req(msg.source, msg.round_number, msg_id)
        else break
        }
}
```

Figure 5.7. Algorithm for pbcast-grb protocol

5.3.3 Pbcast-local protocol

This optimization attempts to perform local error recovery. It uses neighborhood information among group members and works as illustrated in figure 5.8. Assume A, B and C are members of a process group, and B, C are neighbor processes. For instance, if we consider that the process group spreads in a wide area network consisting of local area network components, B and C are located in the same LAN component. Each step in the figure performs the following actions:

1. Process B receives a gossip message from process A, and finds out that it lacks a message M.

2. Process B sends a request for message M to process A.

3. Process B picks a neighbor process C randomly, and informs C that “process A has message M”.

4. If process C lacks M too, it sends process A “multicast M”.

5. If process A didn't multicast message M, it uses multicast to retransmit M.

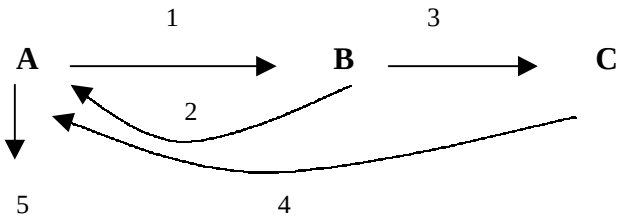


Figure 5.8. An illustration of pbcast-local protocol run

For this optimization, as additional message types, we define *info* and *mcast* messages. Info is the message used to inform a neighbor process about a missing message, as desribed in step-3 above. Mcast is the special request message sent from neighbor process to gossip sender, as described in step-4. These messages contain the following fields:

Info message:
<PT_PBCAST_INFO, process id, sequence number of the requested message>
Mcast message:
<PT_PBCAST_MCAST, sequence number of the requested message>

Figure 5.9 gives the modifications needed for pbcast-local protocol. Two new events are defined that are related to receipt of info and mcast messages.

We model pbcast-local on ns-2 and show that it improves latency distribution of pbcast after FIFO ordering.

Case: Receipt of PT_PBCAST_GOSSIP

```
{
compare my_msg_buffer with msg.msg_buffer
for messages with msg.gossip_retransmit_bit >= 1
        increment their local gossip_retransmit_bit
for each missing message with msg_id { // most recent message first
        request_count ++
```

```
        if (request_count < limit-requests )
                send_req(msg.source, msg.round_number, msg_id)
                pick a neighbor process p randomly
                allocate a packet msg
                msg.type = PT_PBCAST_INFO
                msg.process_id= sender of gossip
                msg.seqno = msg_id
                send(p, msg)
        else break
        }
}
```

New events:

Case: Receipt of PT_PBCAST_INFO

```
{
p = msg.process_id
msg_id = msg.seqno
check my message buffer
if I'm missing message msg_id {
        // send PT_PBCAST_MCAST to process p
        msg.type = PT_PBCAST_MCAST
        msg.seqno = msg_id
        send(p, msg) }
}
```

Case: Receipt of PT_PBCAST_MCAST

```
{
if msg with seqno = msg.seqno is in my_msg_buffer {
msg.request_counter ++
if ((msg.request_counter >= threshold) or
((msg.my_retransmit_bit == 1 ) and (msg.gossip_retransmit_bit >= 1))) {
        send_retrans(groupid, msg.seqno, data message)
        retrans_count ++
        msg.request_counter = 0
        msg.my_retransmit_bit = 0
```

```
        msg.gossip_retransmit_bit = 0 }
}
}
```

Figure 5.9. Algorithm for pbcast-local protocol

5.4 SRM (Scalable Reliable Multicast) Protocol

SRM (Floyd et al., 1997) is a reliable multicast protocol which is designed according to the models of IP multicast group delivery, application level framing (ALF) principle, and the adaptivity and robustness in the TCP/IP architecture design.

IP multicast (Deering and Cheriton, 1990) allows data sources to send to a group without needing any knowledge of the group membership. Basically, IP multicast is a best-effort delivery model and provides no reliability guarantees.

ALF (Clark and Tennenhouse, 1990) is an architectural design principle for data communication. It introduces the integration of the protocol levels from the transport level to the application level. The goal is to provide flexibility and efficiency in the use of the network. However, this leaves the application to include most part of the transport functionality.

SRM follows the core design principles of TCP/IP:

1. It requires only the basic IP delivery model and builts reliability on an end-to-end basis. No change or special support is required from the underlying IP network.

2. In a fashion similar to TCP adaptively setting timers or congestion control windows, SRM algorithms dynamically adjust their control parameters based on the observed performance within a session.

SRM does not provide ordered delivery of messages. The protocol aims to scale well both to large networks and sessions. It exploits a receiver-based reliability mechanism.

5.4.1 Session messages

Session messages for reliable multicast are proposed to:

- Enable receivers to detect the loss of the last packet in a burst,
- Enable the sender to monitor status of receivers.

In SRM, each group member multicasts low-rate, periodic session messages that report the sequence number state for active sources, or the highest sequence number received from every member. In addition to the reception state, the session messages contain timestamps that are used to estimate the distance from each member to every other.

Members also use session messages in SRM to determine the current participants of the session. In addition to state exchange, receivers use the session messages to estimate the one-way distance between nodes. The session packet timestamps are used to estimate the host-to-host distances needed by loss recovery mechanisms.

The timestamps are used in the following manner. Assume that host A sends a session message S_1 at time t_1, and host B receives S_1 at time t_2. Later, at time t_3, host B generates a session message S_2, marked with (t_1, Δ) where $\Delta = t_3-t_2$. Upon receiving S_2 at time t_4, host A can estimate the latency from host B to host A as $(t_4-t_1-\Delta)/2 = ((t_4-t_3)+(t_2-t_1))/2$. This distance estimate does not assume synchronized clocks, it does assume that paths are roughly symmetric.

SRM uses mechanisms similar to XTP, to control the sending of request and repair packets, with the addition that in the SRM design, the random delay before sending a request or repair packet is a function of that member's distance in seconds from the node that triggered the request or repair. These functions are described in the next section.

Repair requests and retransmissions are always multicast to the whole group. A lost packet *ideally* triggers only a single request from a host just downstream of the point of failure.

5.4.2 Loss recovery

Multicast group members detect lost messages by means of gaps in the sequence number. In order to detect losses of the last messages that are sent, SRM uses session messages.

When a group member A detects a message loss, it schedules a retransmission request, and sets a request timer to a value from the uniform distribution on

$$[C_1 * d_{S,A}, (C_1 + C_2) * d_{S,A}] \text{ seconds}$$

where $d_{S,A}$ is member A's estimate of the one-way delay to the original source S of the missing data and C_1, C_2 are request timer parameters. If a member receives a request for the missing data before its own request timer for that data expires, then the member resets its request timer.

When a group member B receives a request from A for a data message that B has a copy, B sets a repair timer to a value from the uniform distribution on

$$[D_1 * d_{A,B}, (D_1 + D_2) * d_{A,B}] \text{ seconds}$$

where $d_{A,B}$ is the B's estimate of the one-way delay to A, and D_1, D_2 are repair timer parameters. If B receives a repair for the missing data before its repair timer expires, then B cancels its repair timer.

5.4.3 Adaptive SRM

As discussed in (Floyd et al., 1997), there is not a single setting for the timer parameters that gives optimal performance for all topologies, session memberships, and loss patterns. For applications where it is desirable to optimize the tradeoff between delay and the number of duplicate requests and repairs, an adaptive algorithm can be used. Adaptive SRM adjusts the timer parameters C_1, C_2, D_1, and D_2 in response to the past behavior of the loss recovery algorithms.

5.5 A Comparison of Basic Pbcast and SRM in terms of Loss Recovery

The anti-entropy protocol is the part of Pbcast that deals with loss recovery. During this phase, each process chooses another process in the multicast group in a random manner, and sends its digest of message history to that process. This happens periodically (i.e. through a sequence of rounds) and concurrently with the transmission of regular multicast messages. On receiving such a gossip message including the submitting process' message history, the receiving process compares the digest with its own message buffer contents. If it lacks some messages that the gossiping process has, then it sends a retransmission request for each missing message, and causes the gossiping process to repair that message by retransmitting it.

We claim that, compared to SRM's loss recovery, Pbcast has much less overhead, and needs less bandwidth. We will now discuss the additional message traffic required for loss recovery:

We assume that Pbcast's round duration for gossip is 100msec, and G is the number of members in the process group. Then, if every process gossips to another process every 100msec, G*10 destinations will receive gossip messages every second.

Periodic session messages of SRM are transmitted every second in multicast mode. This means that, G*G destinations will receive session messages every second, and each process receives G session messages every second.

In the basic Pbcast protocol, if a process detects a message loss, it requires a unicast request and repair message to recover the loss. In the case when one or both of these control messages get lost on a noisy link, additional control messages are required.

In the SRM protocol, on the other hand, in order to guarantee reliable delivery, a process multicasts request message to the whole group when it detects a message loss. Request and repair timers are exploited to suppress duplicate requests and repairs for the same message loss. A corresponding repair message in response to a request is similarly in the form of multicast to the whole group. This feature of SRM's loss recovery mechanism

makes its background overhead and bandwidth requirements to increase as a function of group size, whereas Pbcast's background overhead is scalable and does not increase with the group size.

A detailed analysis and comparison of Pbcast and SRM protocols based on our simulation study are discussed in the next chapter.

5.6 Summary

This chapter starts with describing the ns-2 network simulator used as the underlying environment for our simulation model. We then focus on the design and implementation of basic Pbcast on ns-2, followed by the design and implementation of optimizations to the basic Pbcast. The chapter also gives information on the SRM protocol. Finally, we include a comparison of Pbcast and SRM in terms of loss recovery mechanisms.

6. Simulation Results and Analysis

Based on the simulation model described in chapter 5, we accomplished an extensive simulation study to investigate the behavior and performance of protocols. In this chapter, we describe our simulation study, results and analysis in detail.

6.1 Network and Application Characteristics

Impacts of network environment and characteristics are important when investigating behavior of communication protocols. Simulation models allow gaining power over all parts of the network, and hence lead to better understanding of protocols than the other approaches. Our interest in this simulation study lies in the investigation of behavior and performance of Pbcast protocols developed and their comparison with scalable reliable multicast protocols across various network characteristics and application scenarios. For this purpose, we designed simulations on several network topologies such as star, chain, tree, fully connected and clustered networks. Among these, a tree topology is a general one since it combines characteristics of both chains and stars (Floyd et al., 1997).

Each network in the simulations is constructed from nodes and links. A transmission link can be characterized by its bandwidth and delay. Bandwidth of a link is its information carrying capacity. Link delay defines the time required for a packet to traverse a link. The amount of time required for a packet to traverse a link is defined to be

$$s/b + d$$

where s is the packet size, b (bandwidth) is the speed of the link in bits per second, and d is the link delay in seconds. As discussed in (Guo, 1998), to simulate a wide area network, it is reasonable to set the delay for each link to be 5 milliseconds. Since our aim is to simulate large-scale networks, in our simulations, unless stated otherwise, we set link delay to this value. Each link in our simulated networks is bi-directional, and each direction of a link has the same delay and bandwidth characteristics.

Queues represent locations where packets may be held and dropped. In our simulations, drop-tail (FIFO) queueing support of ns-2 is used for buffer management. Drop-tail implements FIFO scheduling and drop-on-overflow buffer management that is typical to most of today's Internet routers. We use error models that simulate link-level errors or loss of packets. In our simulations, we define packet error rates for various network noise behaviors.

A multicast routing strategy is the mechanism by which the multicast distribution tree is computed. In the Internet, multicast routing trees are constructed using protocols such as Core Based Tree (CBT), Distance Vector Multicast Routing Protocol (DVMRP) and Protocol Independent Multicast (PIM). We use CBT multicast routing strategy support of ns-2 in our simulations.

We construct process group applications on top of networks. Applications sit on top of protocols agents in ns-2 and utilize protocol agents for multicast communication. We use Constant Bit Rate (CBR) data sources for generating data messages to be disseminated group participants. A CBR source generates traffic according to a deterministic rate.

As defined in (Floyd et al., 1997), the density of a multicast session is the ratio of nodes that are members of the multicast group. If many of the nodes in the network are members of the multicast group, this is called a dense session. On the other hand, if the multicast group size is small relative to the network size, this is called a sparse session. We simulate both dense and sparse mode process group applications for the purpose of analyzing its impact. In our simulated process group applications, for simplicity we assume that group membership remains unchanged.

6.2 Performance Metrics

Our analysis work employs performance metrics that we believe are important when investigating the behavior of scalable reliable multicast protocols. We performed simulations of basic Pbcast, pbcast-ipmc, pbcast-grb, pbcast-local, SRM and adaptive SRM protocols. In the simulations of protocols on several network topologies and scenarios, we varied operating parameters such as network size, group size, link error rates and multicast data rates. We analyzed performance metrics such as protocol overhead, throughput, link utilization, inter-arrival distribution, latency distribution and multicast message congestion. The details on how we accomplished analysis of a given metric are given in the corresponding sections of the chapter.

6.3 Simulations of Dense Groups

6.3.1 Tree topology simulations

For this set of simulations, we constructed tree-topology networks with sizes ranging from 20 to 80 nodes. All of the trees have depth 4, and all nodes have a Pbcast or SRM protocol agent attached. The size of the process group in these simulations equals the network size. There's one sender in the group, it's located at the root node of the tree, and a CBR source is attached to the sender, which generates 100 messages/second, and the message size is 210 bytes. We configure network links to have bandwidth of 1.5Mbits each. A low-level system-wide constant noise rate is imposed on the network: each link drops 0.1% of packets passing over it. This loss rate applies to all messages, whether data or control. If a message passes through more than one link to reach its destination, this drop probability accumulates accordingly, since the same noise rate is set on all links.

For the SRM protocol, loss recovery timer parameters are set as follows: Request timers $C_1=C_2=2$, and repair timers $D_1=D_2=\log_{10}G$ where G is the group size.

For each group size and protocol, five distinct simulations were performed with different random seeds. Each simulation lasts 100 seconds during which 10000 messages are multicast to the group by sender.

6.3.1.1 Analysis of overhead and throughput

Background overhead analysis includes measurements for retransmission request messages and repair (or retransmission) messages received by each group member. Duplicate request and repair messages are taken into account in these measurements. Then, mean values are calculated for each simulation. These are not the only forms of overhead on the protocols. For SRM, we omit session messages from this measurement; while for Pbcast, gossip messages are not included. We include such costs in measuring link utilization.

Figure 6.1 shows result graphs where x-axis is the group size and y-axis is the background overhead measurements in the form of request and repair messages received per second, respectively. The results show that, as the network and process group size scale up, the number of control messages received by group members during loss recovery increases linearly for SRM protocols, an effect previously reported in (Liu, 1997; Lucas, 1998; Hanle and Hofmann, 1998; Li and Cheriton, 1998). These costs remain almost constant for Pbcast and Pbcast-ipmc. For the tree topology network simulations, adaptive SRM has a higher overhead compared to SRM with fixed timers. But, later we will see that this is not always the case, and it depends on the topology. Compared to the basic Pbcast protocol, Pbcast-ipmc has a slightly lower overhead in the form of request messages. Since Pbcast-ipmc multicasts repair messages for loss recovery in certain conditions, the repair message overhead increases relative to Pbcast. This is because some group members, that did not actually request a retransmission, will receive a repair, or even duplicate repair messages. However, if a message was missed by multiple receivers, Pbcast-ipmc increases probability of rapid convergence during loss recovery.

In addition, we measured throughput values, that is number of data messages received by each group member. For this set of simulations where system-wide noise rate is low (0.1%), all protocols, namely, Pbcast, Pbcast-ipmc, SRM, and adaptive SRM, guaranteed full reliability. The reliability mechanisms of the protocols overcame data losses, and all receivers delivered 10000 data messages multicast to the group.

6.3.1.2 Analysis of link utilization

We now consider the link utilization of protocols close to the sender. To compute the utilization, we measured the number of bytes on the link outgoing from the sender and incoming to the sender, for messages of all types. Gossip messages for Pbcast and session messages for SRM are included. Thus, this analysis gives an idea on overall bandwidth usage of protocols. We used monitoring facility of the simulator for observing all byte departures on a link.

Figure 6.2 illustrates link utilization of protocols on an outgoing link from sender and on an incoming link to sender versus group size. The units of the y-axis are percentage of link bandwidth used by the protocol messages. For example, since these simulations involve sending 100 210 byte messages per second, or 168kbits/sec, on the outgoing link from the sender, the link utilization required just to send the data would be about 10%. Additional overhead results from request, retransmission and gossip messages in the case of Pbcast. We see that both SRM and adaptive SRM have higher bandwidth consumption compared to Pbcast protocols in both directions of the link being monitored. Link utilization rises rapidly as a function of group size for SRM, while the utilization is lower for Pbcast and also grows more slowly as a function of system size. Note that, at a group size of about 100 members, the sender's link will be saturated and this will trigger packet loss. Pbcast would apparently continue to function in much larger groups.

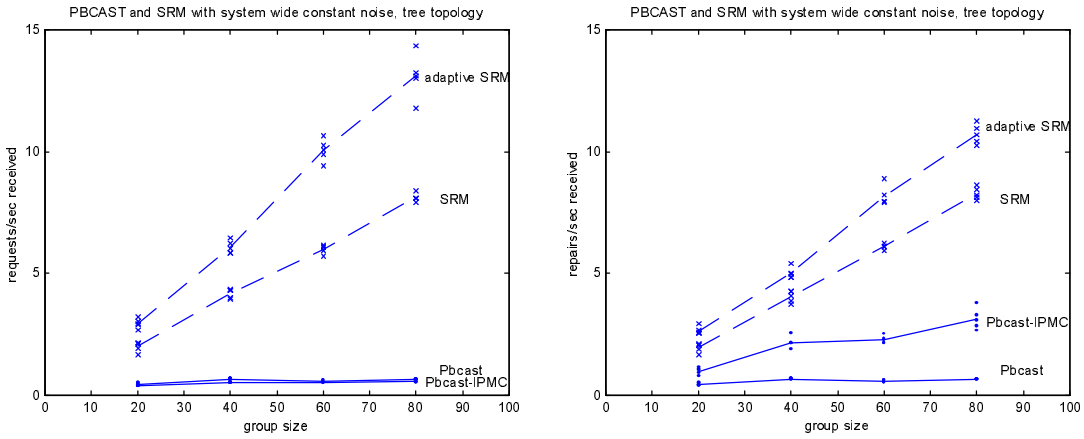


Figure 6.1. Overhead in the form of requests and repairs per second for Pbcast and SRM, tree topology with 0.1% system-wide noise

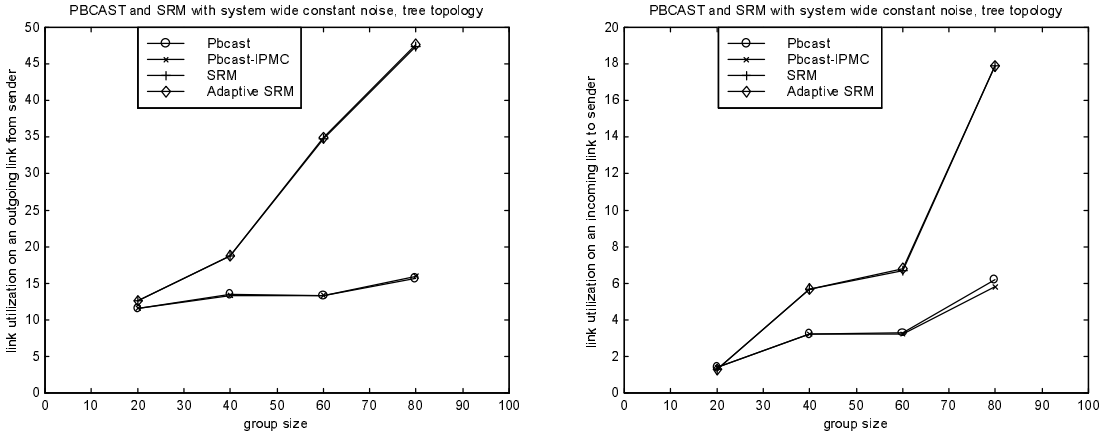


Figure 6.2. Link utilization of Pbcast and SRM on an outgoing link from sender and on an incoming link to sender versus group size

6.3.2 Star topology simulations

This set of simulations investigates the performance of protocols on star topology networks with sizes ranging from 21 to 81 nodes. All nodes, except the center node, have a protocol agent attached. We organize the processes as a star with a single routing node at the center, and the sender and receivers around the periphery. Process group size equals the (network size – 1). Figure 6.3 illustrates a star topology where S denotes the sender, and Ri denotes the receiver i of the process group. In our simulations, there is one sender in the group, a CBR source is attached to the sender that generates 100 messages per second, and the message size is 210 bytes. A system-wide constant noise of rate 0.1% is imposed on the networks. The fixed timer parameters of SRM protocol are set as follows: Request timers C_1=0, C_2=√G, and repair timers D_1=0, D_2=√G where G is the group size.

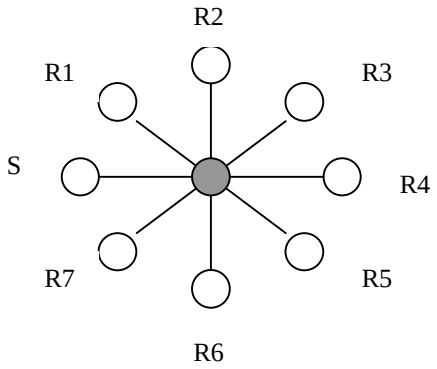


Figure 6.3. A 9-node star topology with a routing node at the center

6.3.2.1 Analysis

Figure 6.4 gives background overhead analysis results for star topology simulations. A star topology in this sense models a local area network where communication latencies are constant and small. The analysis shows that background overhead on group members for both Pbcast protocols is independent of the group size and stays constant with an increase in group size. On the other hand, an increase in background overhead of SRM protocols with group size is observed. Different than the simulations of tree topologies,

for star topology networks adaptive SRM requires less number of repair messages compared to SRM.

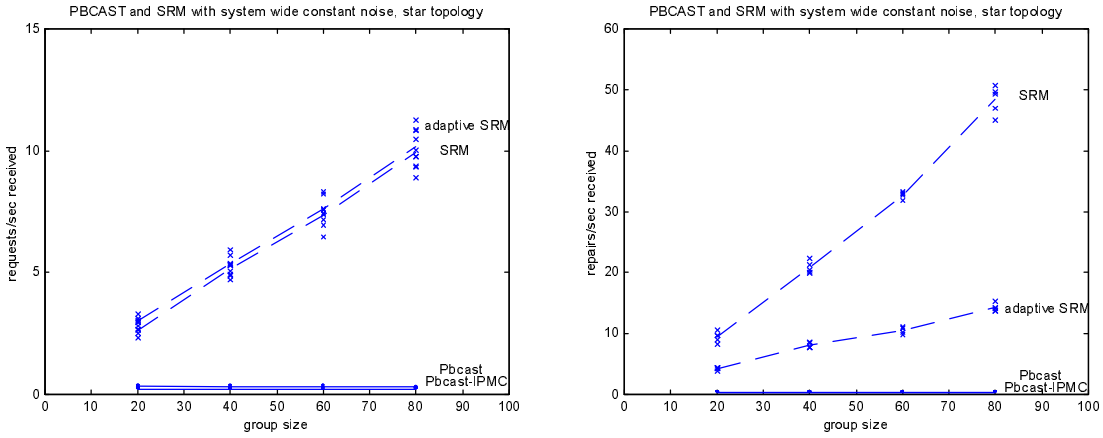


Figure 6.4. Overhead in the form of requests and repairs per second for Pbcast and SRM, star topology with 0.1% system-wide noise

6.4 Simulations of Sparse Groups

6.4.1 Large-Scale Tree Topology Simulations

We explored the impact of scaling the network to a very large size, while keeping the group itself at constant size. In this set of simulations, we constructed large-scale tree topologies consisting of 1000 nodes with tree depth set to 6 and a branching factor of 3. Up to hundred of the 1000 nodes were randomly chosen to be group members in each simulation and that constitutes a sparse session. We set the message loss rate to 0.1% on each link, and ran five simulations with the sender located at the root node injecting 100 210-byte multicast messages per second.

The fixed timer parameters of SRM are set as follows: Request timers $C_1=C_2=2$, and repair timers $D_1=D_2=\log_{10}G$ where G is the group size.

6.4.1.1 Analysis

We analyzed the background overhead of each protocol, and the results obtained are shown in Figure 6.5. To give a sense of the variability of these results, we included error bars showing minimum and maximum values recorded over a set of five runs, using different seeds for the random number generator.

The data is consistent with our findings for the dense tree topologies used in figure 6.1, although the SRM overhead values are somewhat higher. For example, in the 80-member case, the normal SRM request and repair rates rise to about 12 and 18 per second, respectively. This is the double what we saw in a dense session with the same number of group members. Similarly, the adaptive SRM protocol now has overheads of about 20 request and 20 repairs per second, compared to 12 and 10, respectively, in the 80-member dense case. The higher rates are presumably triggered by the higher overall loss experienced as messages flow through the network, since each link has an independent loss behavior. Both pbcast and pbcast-ipmc continue to have low costs. As in the dense case, the impact of multicast retransmissions is evident in a slightly higher rate of repairs.

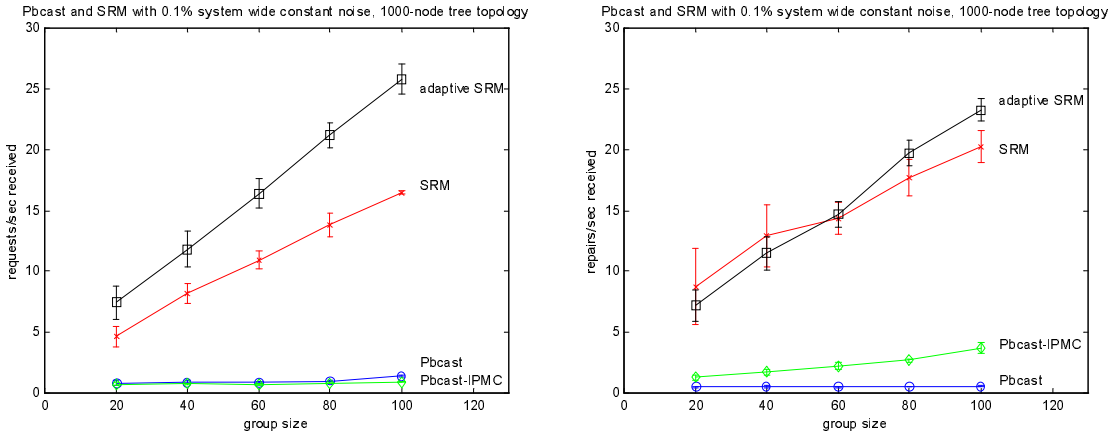


Figure 6.5. Overhead in the form of requests and repairs per second for Pbcast and SRM, 1000-node tree topologies with 0.1% system-wide noise

6.5 Simulations of Larger System-wide Noise Rate

Until now, the network noise rate on our simulations was 0.1%. Now, we increase system-wide noise rate to 1%, which is also a realistic amount that can be observed in real networks. Figure 6.6 shows one of the analysis results giving information on link utilization versus group size on tree topology simulations. Simulation settings are the same as previous tree topology simulations except that system-wide noise rate is increased from 0.1% to 1%. An increase in link utilization of SRM with the group size is observed.

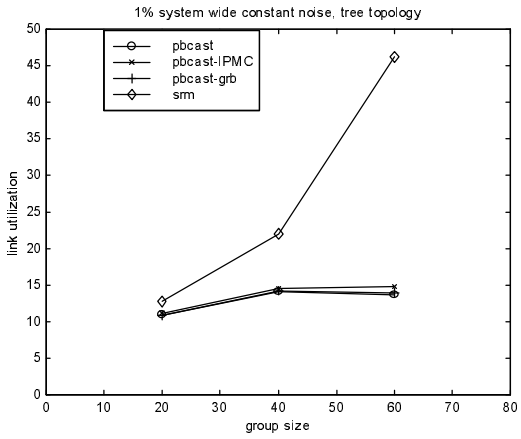


Figure 6.6. Link utilization versus group size for pbcast and SRM protocols

6.6 Inter-arrival distributions

We investigated the inter-arrival distributions of data messages for Pbcast and SRM, and also the effect of an increase in the group size on the distribution. The actual message dissemination rate of the sender in these simulations is 100 messages per second. Therefore, if no message loss occurs, we expect that the inter-arrival time between messages is 0.01 second. When we introduce some noise to the network, there will be some message drops, and loss recovery mechanisms of protocols will generate some data retransmissions to achieve communication reliability. During our simulations, we measured data message arrival times at a typical group member, calculated inter-arrivals

between consecutive messages, and then analyzed individual inter-arrival time values to generate the distribution.

Our simulations use dense tree topologies where every node is a group member, and we define 1% noise on each link. Sender injects data messages at the rate of 100 210-byte messages per second. Figure 6.7(a) shows inter-arrival distributions at a typical receiver for Pbcast-grb on 20, 40 and 60-node tree topologies. As it can be seen, inter-arrival times of data messages are stable with an increase in the group size. Similarly, Figure 6.7(b) shows inter-arrival distributions of SRM on 20, 40 and 60-node tree topologies. Inter-arrival distribution of SRM changes with the group size. In other words, it's not stable. This is mainly due to the higher number of repair messages received by group members during loss recovery and its dependence on group size.

Inter-arrival distribution of a protocol is related to its throughput stability. Previously, stable throughput is not normally considered to be a critical requirement in reliable multicast protocols, but as discussed in chapter 2, we believe that there are a substantial number of applications for which such a guarantee is important.

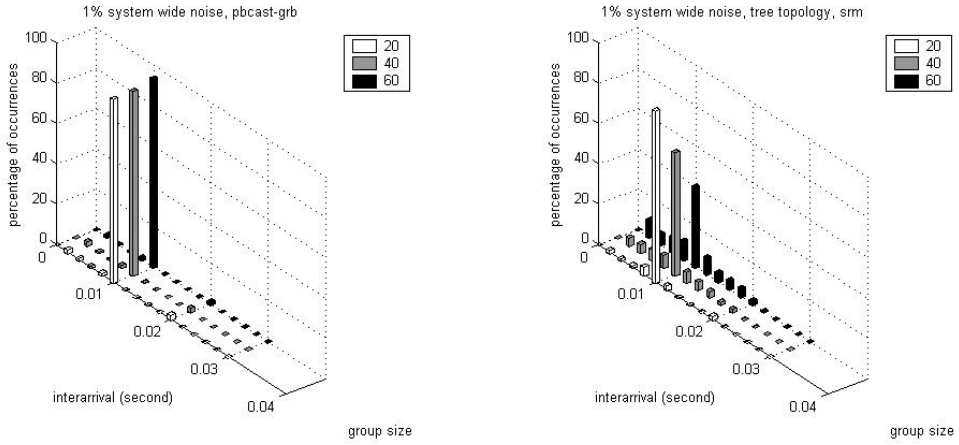


Figure 6.7. Histograms of inter-arrival times of pbcast-grb and SRM with 1% system-wide noise in densely populated tree networks

6.7 Message latency distributions

The latency of a data message at a process is defined as the delay between the time that a message is initially multicast to the group by data source and the time the message is first delivered by the process. There are basically two cases:

1) The message is not exposed to a failure and delivered at the end of best-effort transmission,

2) The message drops because of a failure in the network, and error recovery mechanism takes part to recover the message and makes sure it reaches to the intended destination processes.

In any case, a process can receive duplicate copies of a message, but in our analysis, we do not consider duplicate receipts, and just use first receipt time of a message to calculate its latency.

Our analysis works as follows: We record the times when a message is sent and received by group members. Then based on these data, message latencies for each data message transmitted through the lifetime of the process group are calculated. After that, by using all message latency values, latency distributions are generated. Since Pbcast protocol provides FIFO ordered delivery, we analyze its latency distribution in two forms: Latency distribution at node level, and latency distribution after FIFO ordering. In contrast, since SRM doesn't guarantee ordered delivery, we just analyze its latency distribution at node level.

We accomplished a detailed study of message latency behavior of Pbcast and SRM protocols. In general, results show that on large-scale networks, node-level message latencies of Pbcast protocol is smaller compared to SRM's message latencies. Figure 6.8 shows one of such results where x-axis is latency in seconds and y-axis is percentage of occurrences. Figure 6.8(a) and (c) are the node level latency distribution of Pbcast, and SRM respectively. These simulations were performed on a 500-node tree topology where randomly selected 300 nodes are group members. The sender that is located at the root

node sends with rate 0.01 that is 100 messages per second, and there is a system-wide noise with rate 1%. As it is shown in the figures, a typical receiver delivers messages with lower latencies when Pbcast protocol is used for group communication. As pointed in figure 6.8(c), SRM has a large tail with a maximum observed latency of nearly 500ms, and a group of packets delivered at around 400ms. Overall, SRM has a significant number of packets delivered during the first 100ms and a second broad distribution containing almost 5% of packets, which arrive with latencies between 300ms and 500ms. Notice that the basic SRM distribution is not as tight as the unordered pbcast distribution, which has more than 90% of its packets arriving at the lowest possible latencies. In the case of pbcast, around 2% of packets are delayed and arrive in the period between 200ms and 300ms, with no larger latencies observed.

We also investigated message latencies of Pbcast after FIFO ordering is accomplished. In that case, depending on the message loss rate experienced by the receiver, some percentage of messages are delivered with higher latencies since messages not in order are buffered prior to delivery in order to guarantee FIFO ordering property (Figure 6.8(b)). These higher latencies reflect the cost of waiting for messages to be retransmitted and placing them into the correct delivery member.

We believe these results to be important, at least in settings where steady delivery of data is required by the application. We observe that as SRM is scaled to larger groups, steadiness of throughput can be expected to degrade. We experimented with a variety of noise levels, and obtained similar results, although the actual number of delayed packets obviously depends on the level of noise in the system.

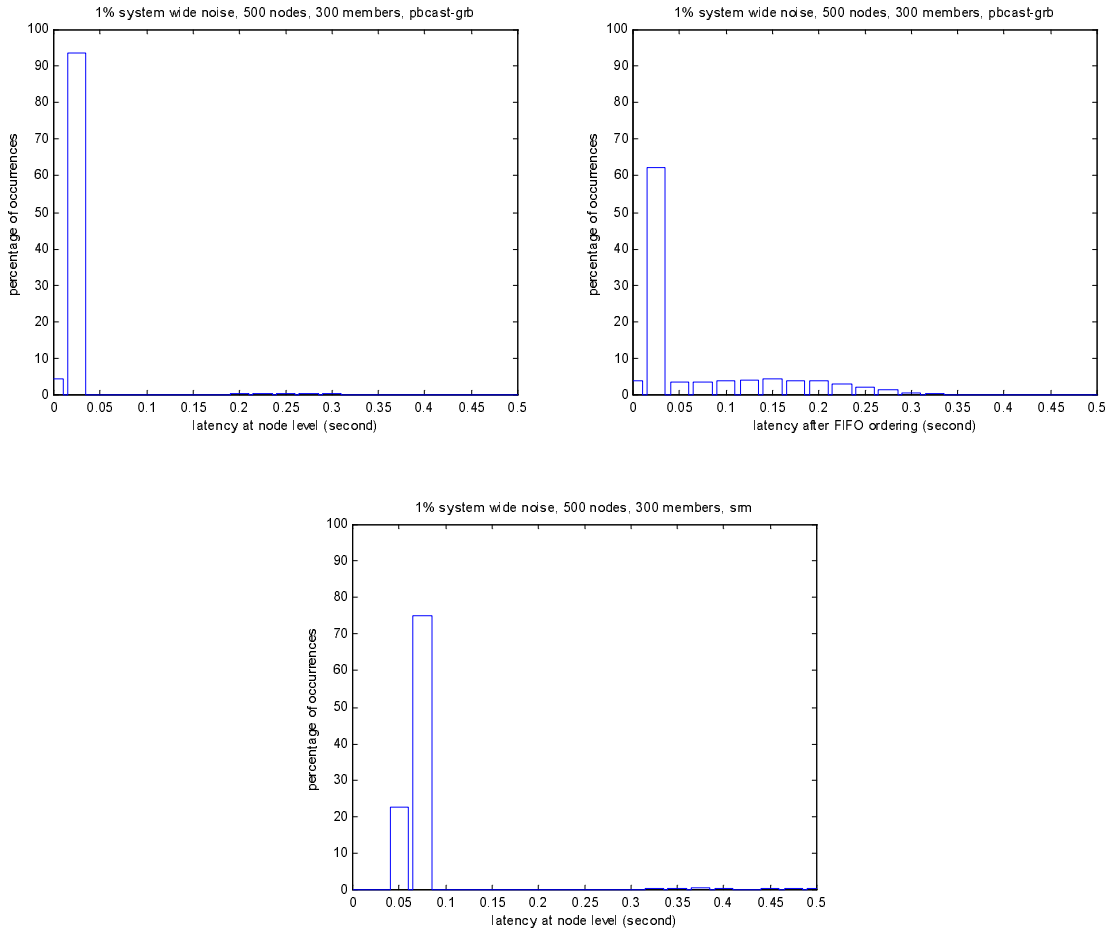


Figure 6.8. Histograms of latencies for pbcast and SRM. a) Latencies for pbcast-grb at node level, b) Latencies for Pbcast-grb after FIFO ordering, c) Latencies for SRM

In our simulations, Pbcast's application level, or after FIFO ordering, latency distributions of different receivers were analyzed to see if the distribution changes depending on the receiver's distance from the source node. In other words, our interest is in the impact of distance from the sender on latency. The results show that application level latency distribution of Pbcast is independent of the receiver's distance.

Figure 6.9 shows one set of graphs illustrating this outcome. The simulation settings are as follows: The network consists of 20 nodes with linear (chain) topology where first

node is the sender spreading 100 messages per second to the process group, and there is 1% noise on the outgoing link from the sender. Remaining members are receivers. Each link has a transmission delay of 5msec. We analyzed application level latency distributions of all receivers and observed that the distribution is basically the same for the receivers. Naturally, beginning of latency interval changes depending on the total link delay of the receiver from the sender. The figure shows latency distributions of four such receivers that are with distance 2, 8, 14, and 18 hops from the sender.

Theoretical analysis of pbcast (Birman et al., 1999) suggests that the distribution should not change, and this is confirmed by our simulation model. The only effect is to introduce a small offset to the distribution, corresponding closely to the network delay itself. We obtain the same results in other network topologies.

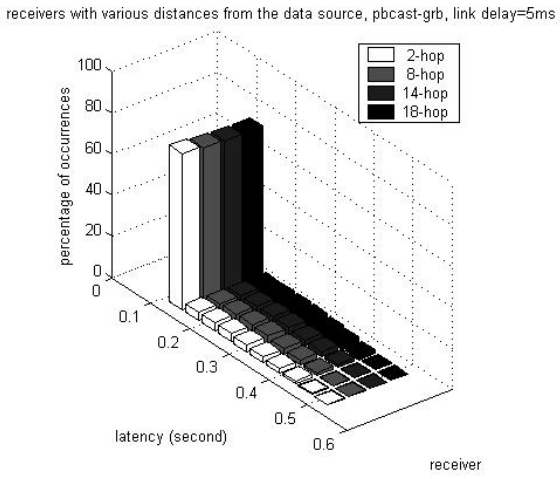


Figure 6.9. Latency distributions of pbcast-grb for receivers at various distances from the data source

6.8 Simulation of Clustered Network Topologies

Until now, our simulations have focused on the impact of randomized message loss on the performance of Pbcast and SRM protocols. Other scenarios might be local area networks connected by long distance links and networks where routers with limited bandwidth connect group members. Such configurations are common in today's networks.

6.8.1 Clusters Connected by a Noisy Link

In this scenario, we simulate a clustered network with 80 nodes as sketched in Figure 6.10. The network consists of two 40-node fully connected clusters, and a single link connects those clusters. All nodes have Pbcast-grb, SRM or adaptive SRM agent attached that forms an 80-member process group. Sender is located on the first cluster, and it generates 100 multicast messages per second. There is 1% intracluster noise formed in both clusters, and a high noise with rate 50% is injected on the link connecting the clusters. This intercluster noise behavior leads to a condition where with 50% probability every message transmitted between clusters will drop. We then explored the latency characteristics of a receiver on the second cluster.

Figure 6.11 shows latency distribution of Pbcast-grb at node level, or without FIFO ordering and after FIFO ordering. Figure 6.12 shows latency distribution of SRM and adaptive SRM at node level. The latency distribution of Pbcast-grb remains relatively tight, in the range between 0 and 1000 millisecond. Unlike the distributions analyzed in the tree topology networks, most messages are now affected by a delay. This is probably due to the high loss rate we imposed on the link connecting two clusters. Latency distributions of SRM exhibit long delays, particularly for the adaptive SRM, which has a significant number of long delayed packets. Note that, the 'spike' seen in the adaptive SRM distribution at latency equal to 5 seconds occurs because all packets with latencies greater than or equal to 5 seconds are counted in this single 'bin'. Thus, in this configuration, both SRM and adaptive SRM deliver some messages with very long delays of many seconds. Particularly, in the adaptive case about 5% of all data messages are delayed by 5 seconds or more before delivery. On the other hand, Pbcast-grb delivers

all data messages within 1 second and hence can be seen as offering relatively steady data throughput in networks with this configuration.

The results suggest that, for Pbcast protocol, message latencies of receivers suffering from high message drop rates are better even after FIFO ordering relative to SRM and adaptive SRM protocols.

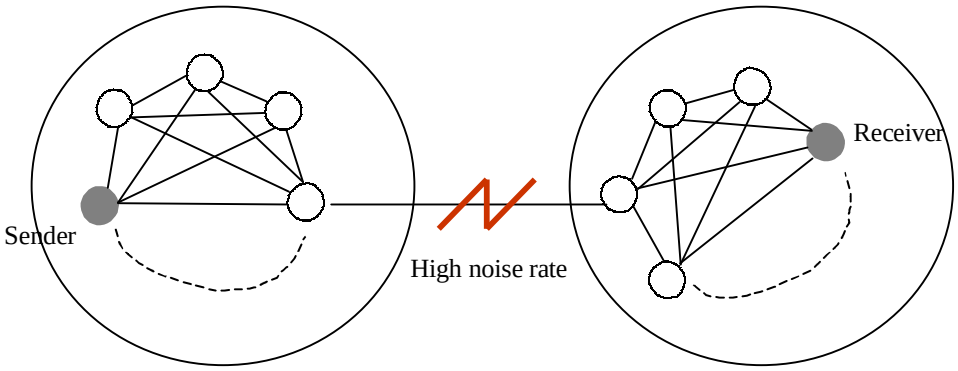


Figure 6.10. Two clusters connected by a noisy link

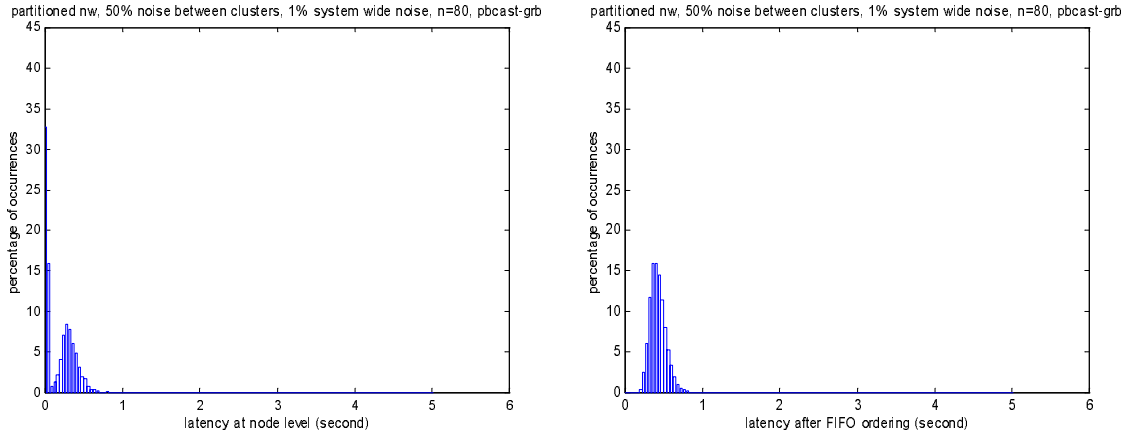


Figure 6.11. Delivery latencies of Pbcast-grb before and after FIFO ordering in a two-cluster network

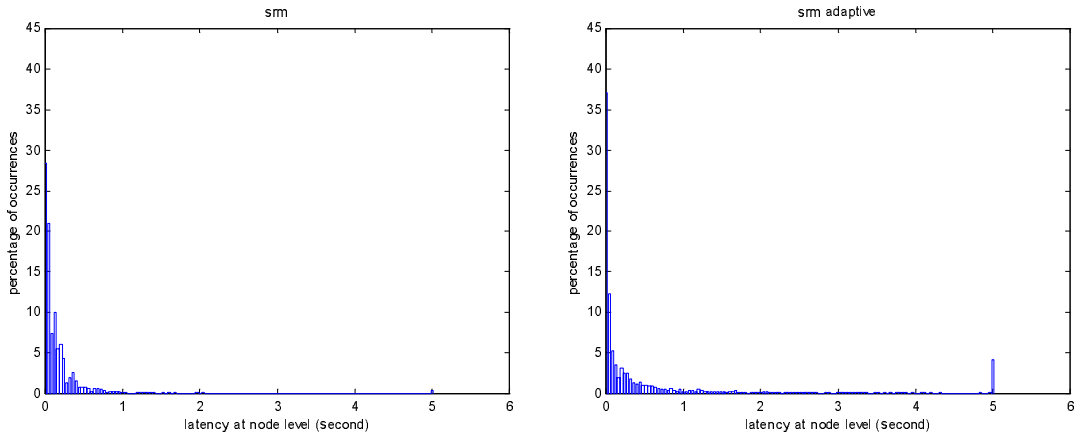


Figure 6.12. Delivery latencies of SRM and adaptive SRM in a two-cluster network

6.8.2 Limited Bandwidth on Router

In order to see the effect of limited bandwidth on a router, we constructed tree topology simulations running on 20, 40, 60, 80 and 100-node networks. Figure 6.13 illustrates such a network. The root node behaves as a router, and links to the router have limited bandwidth of 1500Kbits compared to the other links of the network that all have bandwidth of 10Mbits. System-wide constant noise rate is set to 1%. As shown in the figure, one of the nodes on the left sub-tree is sender which sends 100 multicast messages per second, and the message size is 1000 bytes. Therefore, the sender disseminates 800Kbits per second to the network that is around the half of the capacity of the limited bandwidth. All the other nodes are receivers. In these simulations, we analyzed the background overhead and latency distribution of the particular receiver on the right sub-tree that is illustrated in the figure.

Figure 6.14 shows background overhead analysis of Pbcast and SRM for this scenario. A dramatic increase in especially request message traffic of SRM is observed for large group size at which limited bandwidth capacity starts to show its effect on SRM's control traffic requirements. The reason is that the router becomes saturated and consequently the loss rate near the router rises. The rate of requests for Pbcast-grb remains nearly constant,

and the growth in repairs is consistent with the size of the group and the high noise rate used in this scenario.

For 20 and 40-node network simulations, message latency distributions of Pbcast-grb and SRM receiver resemble each other. But, as network and group size increases, we note that communication requirements of SRM start to exceed capacity of the limited bandwidth, and this dramatically affects latencies of messages received by group members on the right sub-tree. We include analysis results for 100-node topology in Figure 6.15 where the effect of limited bandwidth on a router for SRM protocol can be seen clearly. Figure 6.15(a) and (b) show latency distribution of Pbcast-grb at node level and after FIFO ordering for the particular receiver. As it can be seen in Figure 6.15(c), for the SRM case, a large percentage of messages are delivered with high latency values going up to 15 seconds.

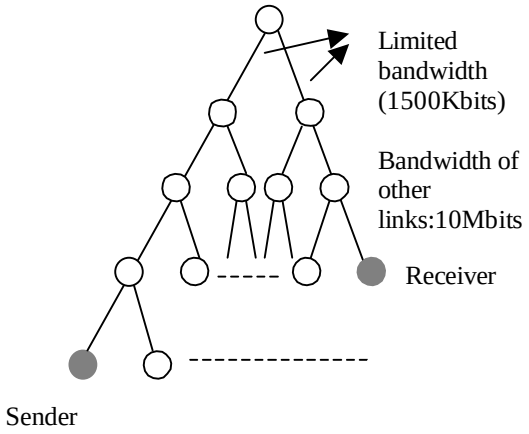


Figure 6.13. A tree topology with a router with limited bandwidth at root

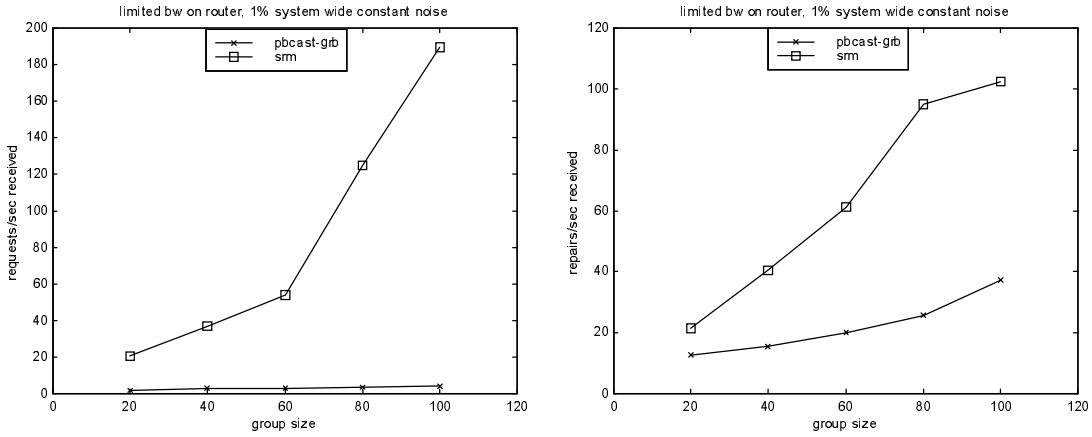


Figure 6.14. Request and repairs in tree topologies with limited bandwidth on root links

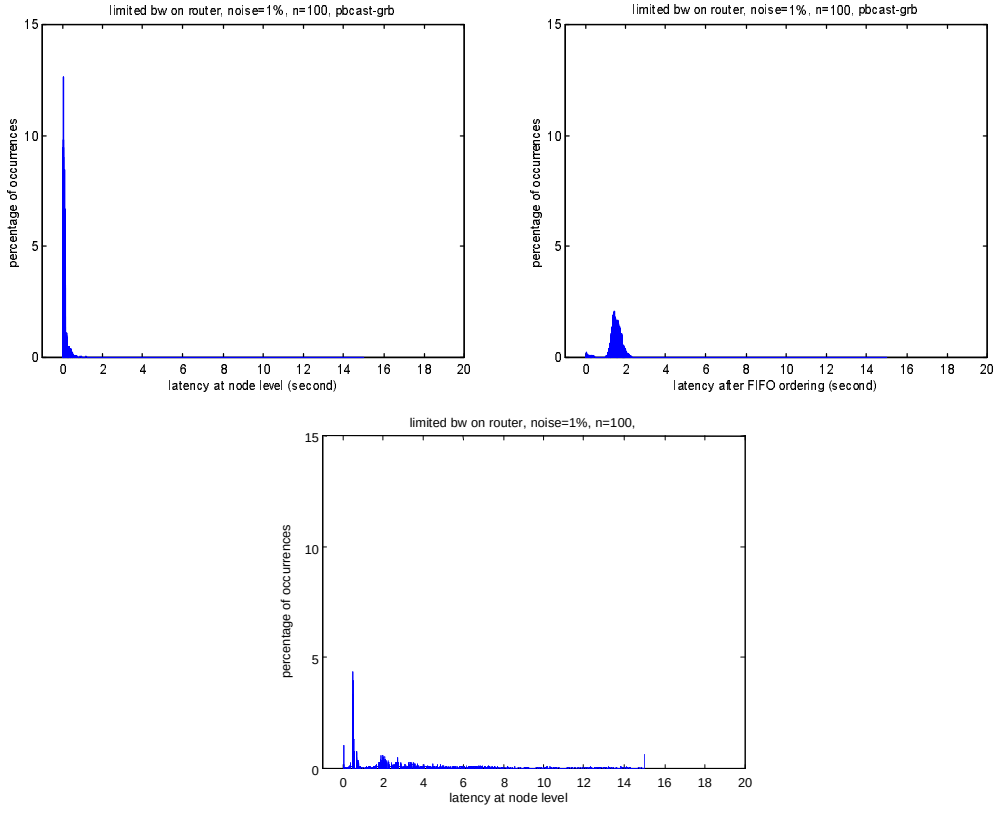


Figure 6.15. Delivery latencies for pbcast and SRM in 100-node tree topologies with limited bandwidth on root links. a) Latencies for pbcast-grb at node level, b) Latencies for Pbcast-grb after FIFO ordering, c) Latencies for SRM

6.9 Impact of Pbcast-local on Application Level Latencies

In this set of simulations, we constructed a 60-node tree topology network, and injected 10% constant noise on one outgoing link from sender, where the sender is located at the root node. The other nodes are receivers. Then, we picked a receiver that is exposed to some message losses because of the noise on the network link. On this simulation setting, we run both Pbcast-grb and Pbcast-local under the same conditions. Then, we analyzed the latency distributions of the receiver for both protocols after FIFO ordering.

Figure 6.16 shows latency distributions where Pbcast-local has a notable improvement on the latency characteristics of Pbcast protocol. As described in chapter 5, Pbcast-local uses neighborhood information among group members and attempt to accomplish local recovery. As our analysis results indicate, this optimization improves latency of data messages.

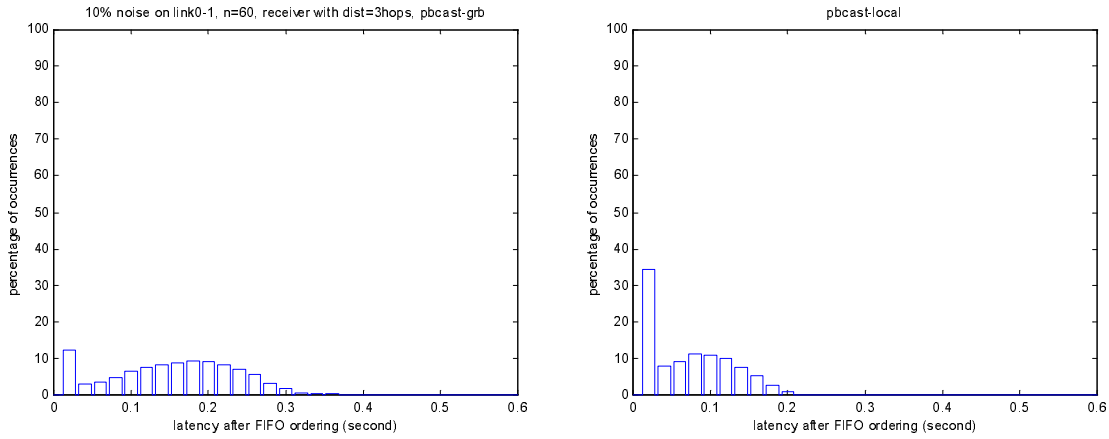


Figure 6.16. Application level latency distributions of a) Pbcast-grb, b) Pbcast-local

6.10 Conditions Causing Multicast Message Congestion

We investigate the conditions that cause multicast messages to begin congest when using SRM and pbcast-grb. The network topology in this simulation study is a 2-cluster, 80-node network. Each cluster consists of fully connected 40 nodes. There is a single link connecting two clusters that has higher noise and link delay characteristics, behaving like a long distance link. The delay of links inside clusters is set to 5ms, while the link delay between clusters is 30ms. There is 1% low noise injected on the links inside clusters. We formed an 80-member process group on this topology where there is a group member on each node. The sender is located on the first cluster. During this study, we varied two operating parameters, namely multicast message rate of the sender and inter-cluster noise probability. The multicast message rate is set to 25, 50 and 100 messages per second, and the inter-cluster noise probability is set to 10, 20, 30, 40 and 50% for different simulations.

We analyzed the behavior of a receiver process on the second cluster. We observed the change in degree of interference while load and error rate increase. *Degree of interference* is defined to be the percentage of data messages with latencies greater than normal message delay. *Normal message delay (nd)* for a particular receiver is defined to be

$$1 / msgrate + ld$$

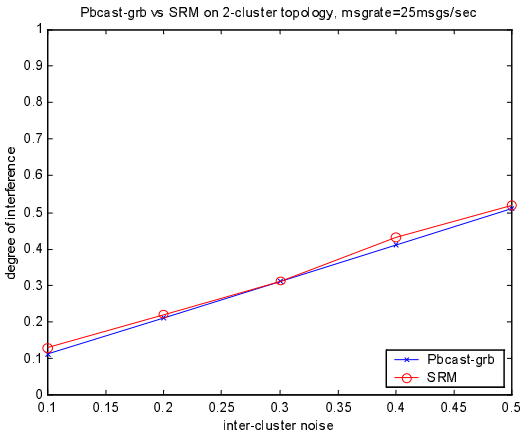
where *msgrate* is the message rate of the sender, and *ld* is the total link delay from sender to the receiver. For example, if the message rate of the sender is 25 msgs/sec and the total link delay from the sender to a particular receiver in the group is 40ms, then nd = (1/25) + 0.04 = 0.08sec. We assume that messages received with latencies greater than 0.08sec are delayed because of the interference, and analyzed the percentage of data messages experiencing this delay.

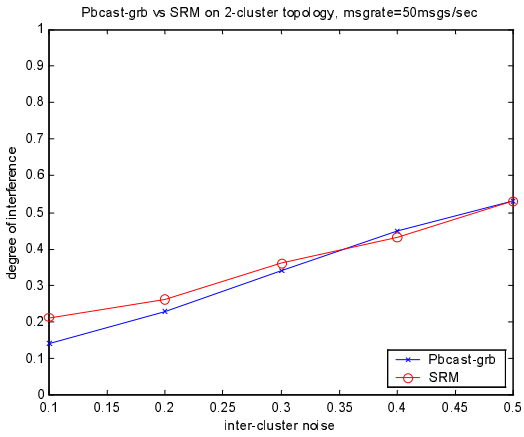
Figure 6.17 illustrates the change in the degree of interference as the link error rate between the sender and the receiver increases. The x-axis shows the inter-cluster noise rate and the y-axis shows the degree of interference measured for both pbcast-grb and

SRM. Message rate of the sender is 25, 50 and 100 messages per second for figure 6.17(a), (b) and (c) respectively.

Figure 6.18 shows the change in the degree of interference as the load rate increases. The x-axis is the multicast message rate of the sender, and the y-axis is the degree of interference. Inter-cluster noise rate is 10% for figure 6.18(a), and 20% for figure 6.18(b).

We observed that for most of the simulation results, the number of data messages experiencing this interference is greater for SRM protocol compared to pbcast-grb. Error rate and load increase in the network affect the interference parameter. As the load increases, the difference between both protocols becomes more apparent. Another observation about the reliability of the protocols not shown in the figures is that when inter-cluster error rate exceeds 40%, the SRM receiver starts to fail in its recovery phase and lose some data messages, while no message loss was observed for pbcast.





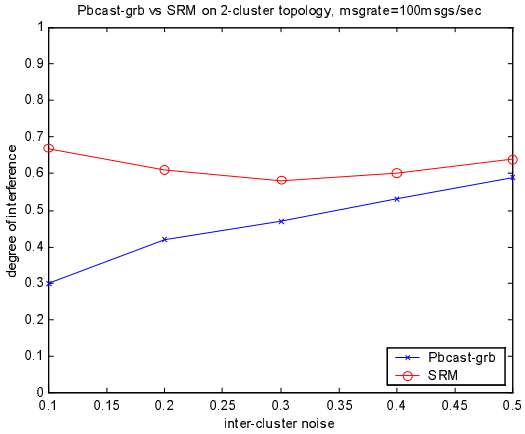


Figure 6.17. Inter-cluster noise rate vs. degree of interference for Pbcast-grb and SRM. a) Multicast message rate equals 25msgs/sec, b) Multicast message rate equals 50msgs/sec, c) Multicast message rate equals 100msgs/sec

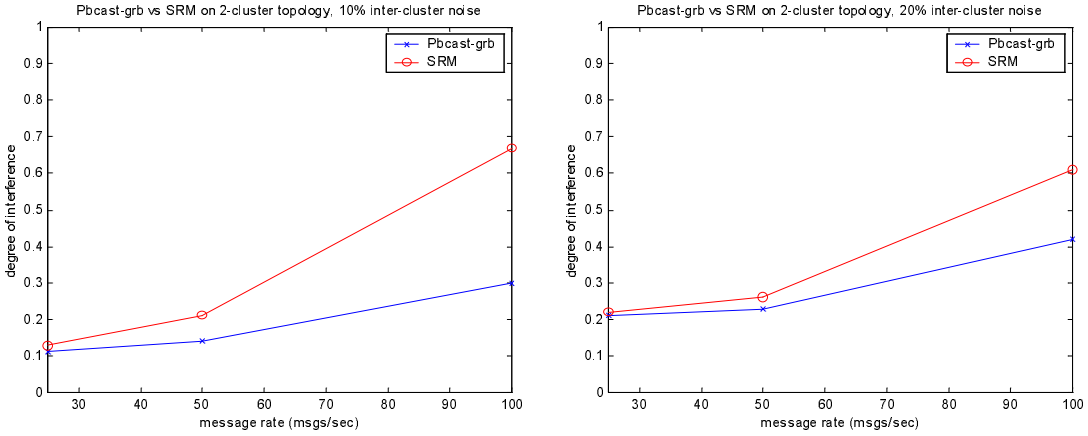


Figure 6.18. Multicast message rate vs. degree of interference for Pbcast-grb and SRM. a) Inter-cluster noise rate is 10%, b) Inter-cluster noise rate is 20%

6.11 Discussion

Our simulation study yields some general conclusions about the behavior of scalable reliable multicast protocols in large systems, some specific conclusions about the relative advantages and disadvantages of SRM and Pbcast, and also the limitations associated with each protocol.

Our work points to a number of limitations of the SRM protocol. We observe that SRM can generate high rates of overhead on a network with lossy links, even if the loss rate is low. Some prior work points this effect as well (Liu, 1997; Lucas, 1998; Hanle and Hofmann, 1998; Li and Cheriton, 1998). SRM protocol overhead is in the form of requests and retransmissions sent using multicast and hence seen by significant numbers of processes. As the network size scales up, overhead rate increases. As a result, overall bandwidth requirement of the protocol grows as well.

As it is shown in the analysis of clustered network topologies, high overhead rate can cause routers in a wide area network to become saturated easily. Another problem is evident in the latency distributions of SRM in clustered networks with a noisy connecting link, or limited bandwidth behavior near the connecting router. We believe, these are not unlikely situations, in fact they are common in typical LAN configurations with a WAN link between two LANs. Under such conditions, we analyze that delivery latency for SRM goes very high values relative to the actual source-to-destination network latency. As shown in figure 6.15(c), worst-case latency value of 15 seconds or more is measured where the actual source-to-destination network latency is 35ms. A significant percentage of SRM packets experience long delays, and this may cause many applications to be forced to buffer very large amounts of data. For applications in which data freshness is important, this would seem to be a real drawback for the protocol. Furthermore, under the conditions analyzed in section 6.10, increases in the application data rate can cause high rates of multicast messages to begin congest.

Analysis results for Pbcast protocol are more positive under the same network conditions. But, we observe some limitations as well. In general, where SRM shows severe overhead growth, Pbcast sometimes shows moderate overhead growth. Consequently, one can

criticize that Pbcast also faces some scalability limits, but our analysis shows that overheads seem unlikely to emerge until a network grows quite large. This is evident in the results of large-scale topologies with thousand nodes (figure 6.5).

Another issue about Pbcast is the gossip load on centralized links in multi-clustered networks. If the capacity of a central link is exceeded, this will trigger a high rate of loss on the corresponding link. This would impact other applications sharing the network. As a remedy for this issue, hierarchical gossip mechanisms are explored in the Spinglass project implementation of Pbcast and this is an area for further study. Initial results for hierarchical gossip strategy are discussed in (Ozkasap et al., 1999.a).

Our studies show that Pbcast is a better behaving reliable multicast protocol than SRM in the network settings that are considered. Our findings are based on the following facts. The issue is about the impact of random low-probability events on the behavior of scalable reliable multicast protocols. SRM protocol uses timers and suppression mechanisms that are parameterized according to the network characteristics. These can be viewed as probabilistic mechanisms for overcoming data loss. Introducing *system-wide* link loss rates, even at low levels like 0.1%, apparently defeats SRM's assumptions as the network grows large. The basic hypothesis of SRM is that most multicast data messages will be successfully delivered by IP multicast and basic forms of data loss would be entirely local or regional. For instance, a sub-tree in a tree topology network drops a message, but no other sub-tree does so. Then, the loss recovery would be overcome locally by utilizing timer-based recovery mechanisms. But, in real network settings, processes in both sides of a large spanning tree could experience data loss. Timer mechanisms for SRM are supposed to inhibit duplicate retransmission requests. As the network scales up, processes experiencing loss in both sides of the network would be further away from each other and there would be more processes experiencing the loss in between. SRM mechanisms make no provision for this effect. It would be more likely for both processes to request a retransmission at the same time. Likewise, it becomes likely for multiple processes to respond to a single request. This is particularly evident when all participants are equidistant. Our analysis results for the star topology simulations show this effect (figure 6.4).

We observe a fundamental advantage of Pbcast relative to SRM. Unlike SRM, Pbcast is based on weak assumptions. Gossip mechanisms are highly randomized, and random data loss is attacked by randomized gossip repair. The exponential convergence of gossip towards full diffusion of information in the network is the benefit for loss recovery, and leads low protocol overhead. When IP multicast is used occasionally for retransmission as Pbcast-ipmc and Pbcast-grb do so, multicast data reaches most participants.

6.12 Comparison with Prior Work

We believe that our study is the first investigating the stability of latency distributions in scalable reliable multicast settings. We investigated the behavior of Pbcast protocol together with a well-known scalable reliable multicast protocol SRM, under a variety of network settings. However, there are certainly some prior work that criticized SRM, and some studies have also proposed similar protocols with better local repair strategies giving faster convergence with less overhead. First among these was a study by one of the SRM developers who also proposes some extensions to improve the behavior of the protocol (Liu, 1997). Another study is by (Lucas, 1998), that identifies similar issues with SRM.

Li and Cheriton (1998) have introduced a reliable multicast protocol called OTERS that provides low recovery latency and low recovery traffic levels while requiring some additional network support. Their work uses ns-2 to compare OTERS with SRM and TMPT. They simulate transit-stub network topologies with sizes 100 and 600, with group sizes 10 and 60 respectively, and link error rates of 0.5%. The simulation study focuses on the analysis of recovery latency and traffic load for loss recovery. Among their findings, the authors note that SRM can perform poorly. However, they did not encounter anything as extreme as what we saw in our clustered network simulations.

Another study (Hanle and Hofmann, 1998) uses ns-2 to evaluate performances of SRM, MFTP (Multicast File Transfer Protocol) and MFTP/EC (MFTP with Error Correction). They performed analysis for two different test networks, one with light and the other with intensive background traffic feature. Network sizes for two set of simulations were 726 nodes and 680 nodes where 50 of the nodes are receivers. The results are similar to our

findings for SRM's link utilization. As stated in the paper, the behavior of SRM is due to the fact that, it can easily run into a situation in which multiple repair packets are multicast in response to a single retransmission request. In general, there is a trade-off in SRM between duplicate packet flow and loss recovery speed.

Our simulation study differs from the prior work in the following ways:

- We introduced system-wide constant link loss rate to the networks.
- Link noise affects both data and control messages. For example, in the simulations of the SRM paper (Floyd et al., 1997), it is assumed that only data packets drop during the simulation, but this is not realistic.
- Multicast data is generated in steady rates during the simulations.
- Several realistic scenarios such as routers with limited bandwidth, clusters connected with high noise links, and also various network topologies are considered.
- In addition to background traffic and link utilization analysis, message latency, inter-arrival distributions, throughput characteristics of protocols, conditions causing multicast message congestion are analyzed as well.
- Large-scale networks consisting of thousand nodes are simulated.

6.13 Summary

This chapter first describes network and application characteristics of our simulation study. In the simulations, we analyzed performance metrics such as protocol overhead, throughput, link utilization, inter-arrival distribution, latency distribution and multicast message congestion on several network topologies and group application scenarios. The chapter explains each of these simulations, results and analysis in detail. A discussion on the general results of the simulation study developed in this thesis is included in the chapter, followed by a comparison of the study with prior work.

7. Efficient Buffering

Traditionally, reliable multicast protocols suffer from large buffering requirements. Group participants have to buffer messages and buffer sizes grow with the number of participants. We describe a technique that allows such protocols to reduce the amount of buffering significantly. The idea was first suggested by Robbert van Renesse, and in this thesis we conducted a simulation and analysis study to validate the effectiveness of the optimization (Ozkasap et al., 1999.b). This chapter gives details of this study.

7.1 Model

We assume a group of processes or members communicating with an epidemic reliable multicast protocol such as used in the Clearinghouse domain name service (Demers et al., 1987), refdbms (Golding et al., 1994), Bayou (Petersen et al., 1997) and Bimodal Multicast. In our simulation study, we use bimodal multicast as the underlying multicast communication structure. Each member of the process group is uniquely identified by its address. We consider that each member has an approximation of the entire membership of the group. It is not required that the members agree on the membership. A scalable membership protocol such as the one proposed by (van Renesse et al., 1998) is sufficient to provide membership information. We consider a fail-stop model of processes. Malicious failures are not considered in this model. Recovery of a failed process is modeled as a new process joining the membership.

Related to link failures, we assume two kinds of message loss, namely, send omissions and receive omissions. Initially, we assume that receive omissions are independent from receiver to receiver and message to message, and occur with a small probability P_{loss}, and there are no send omissions.

As mentioned before, members communicate via a reliable multicast protocol that aims to provide all-or-none delivery of multicast messages. In general, such protocols run in three phases:

1) An initial unreliable multicast dissemination attempts to reach as many members as possible.

2) An error recovery phase detects message losses and recovers lost message via retransmissions.

3) A garbage collection phase detects message stability and releases buffer space.

Most reliable multicast protocols use a combination of positive and negative acknowledgement messages for the last two phases. Epidemic multicast protocols achieve the all-or-none guarantee with high probability by means of gossiping. Garbage collection is accomplished by having members keep messages in their buffer for a limited time. Members garbage collect a message after a time at which they can be sure, with a specific high probability, that the gossiping has disseminated all messages that were lost during the initial multicast dissemination. This time grows as O(logn) where n is the size of the membership as the corresponding member observes it (Birman et al., 1999).

7.2 Basic optimization

The proposed technique optimizes buffering by only buffering messages on a small subset of group participants, while spreading the load of buffering over the entire membership. The subset has a desired constant size C. Failures and other randomized effects, such as randomness in the outcomes of the hash function and inconsistencies due to the approximation of the group membership information, cause messages to be

buffered on more or fewer than C participants. The subset is not fixed but randomized from message to message in order to spread the load of buffering evenly over the membership. We assume that each message is uniquely identified. For example, the tuple (source address, sequence number) identifies a message. We use hash function H: bitstring $\rightarrow$ [0..1] to map tuples of the form <message identifier, member address> to numbers between 0 and 1. This hash function has a certain fairness property. For a set of different inputs, the outputs should be unrelated. There are a number of choices for this purpose. Cryptographic hashes are ideal, but too CPU intensive. CRCs (Cyclic Redundancy Checks) are cheap, but the output is too predictable. When given 32-bit numbers 0,1,2,... as input, the output of CRC-16 is 0,256,512,etc. We propose a hash function that is cheap and appears fair. The function and its properties are described in section 7.3.

A multicast message sent to the process group is buffered on a small set of members. Suppose that a member with address A has a view of the membership of size n. Upon receiving a message with identifier M, member A buffers the message if and only if H(<M,A>)*n < C. We call a member that buffers M, the bufferer of M. If the function H is fair, n is correct and there is no message loss, the expected number of bufferers for message M is C. For a set of messages M1, M2, ..., the messages are buffered evenly over the membership.

Figure 7.1 illustrates the buffering technique on a simple scenario. Suppose, there is a four-member process group, n=4 and C=1. A1, A2, A3 and A4 are group members, and the time advances from left to right in the figure. The scenario proceeds as follows:

1. Member A1 multicasts message with identifier M1, and all group members deliver M1. Upon receiving M1, each member Ai calculates H(<M1,Ai>) and buffers M1 if and only if H(<M1,Ai>)*n < C. We assume that H(<M1,A1>)=0.2, H(<M1,A2>)=0.8, H(<M1,A3>)=0.3 and H(<M1,A4>)=0.9. Therefore, since only H(<M1,A1>)*n < C, member A1 is the bufferer of the message M1.

2. Member A3 multicasts message with identifier M2. Due to a transient communication or process failure, member A1 fails to receive M2. The other group members deliver

M2. We assume that H(<M2,A1>)=0.5, H(<M2,A2>)=0.4, H(<M2,A3>)=0.6 and H(<M2,A4>)=0.1. Therefore, since only H(<M2,A4>)*n < C, member A4 is the bufferer of the message M2.

3. Member A1 multicasts message with identifier M3, and all group members deliver M3. We assume that H(<M3,A1>)=0.8, H(<M3,A2>)=0.2, H(<M3,A3>)=0.4 and H(<M3,A4>)=0.7. Therefore, since only H(<M3,A2>)*n < C, member A2 is the bufferer of the message M3. By means of the loss detection and recovery mechanism of the underlying reliable multicast protocol, member A1 detects that it lacks message M2. Then, A1 computes H(<M2,Ai>) for each group member Ai. If H(<M2,Ai>)*n<C for a given member Ai, then Ai is the bufferer of message M2. By means of this technique, A1 determines that A4 is the bufferer of M2, and sends A4 a request for retransmission of M2. Upon receiving the request, member A4 retransmits M2 from its message buffer.

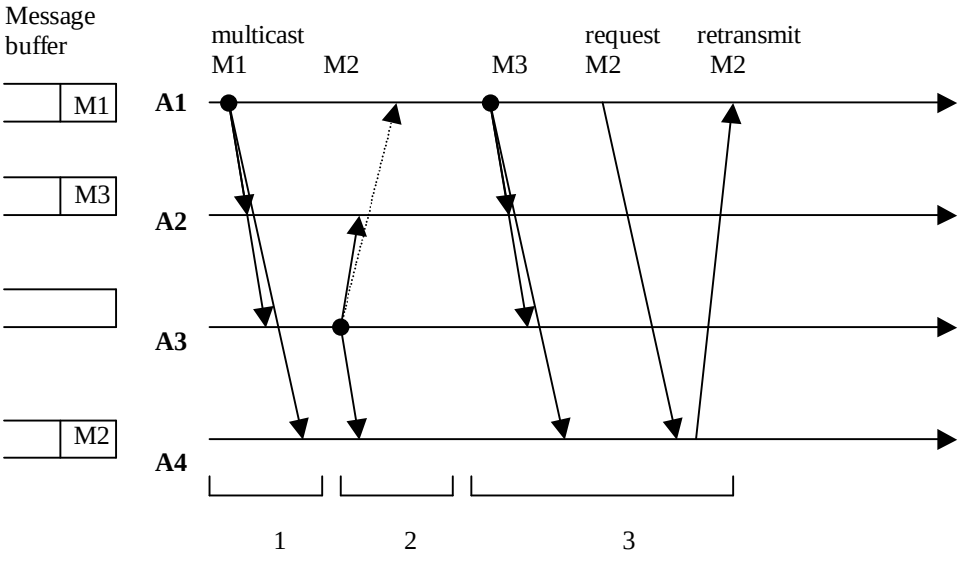


Figure 7.1. Illustration of the buffering technique

We do not require that members agree on the membership. An approximation of the entire membership is sufficient. If members agree on the membership, then any member

can calculate for any message which members are the bufferers. If members have approximate membership information slightly different from each other, it is possible that they disagree on the set of bufferers for a message. However, keep in mind that this is completely independent of the decision to buffer a received message, which is deterministic once the message reaches a given process. In particular, the sets of bufferers calculated by different members mostly overlap. If C is chosen large enough, the probability that all computed bufferers will turn out to have failed to receive a message due to link or process failures is small. When calculating this probability, we assume that every member agrees on the membership and the number of members is n. We consider an initial multicast as successful if it is received by all members, or if it is received by at least one bufferer. If at least one bufferer of an initial multicast receives the multicast, the bufferer would keep a copy of the multicast message in its message buffer, and (by using this copy) loss recovery mechanism of the reliable multicast protocol would be able to disseminate the message to the members that lack the message. Thus, the probability of success of an initial multicast is the sum of the following two independent probabilities.

P_1: no members are bufferers, but all members received the initial multicast

P_2: there is at least one member that is a bufferer and that received the initial multicast

We can calculate P_1 as follows. Since we assume the function H is fair, 'being a bufferer' is an independent event with probability C/n. Also, the message loss due to receive omissions are assumed to be independent with probability P_{loss}.

$$P_1 = ((1 - C/n)(1 - P_{loss}))^n$$

We can calculate P_2 as follows.

$$\begin{aligned} P_2 &= P(\exists \text{ a bufferer that receives M}) \\ &= 1 - P(\text{all processes are not bufferers or lose M}) \\ &= 1 - P(\text{a process is not a bufferer or loses M})^n \end{aligned}$$

$$= \quad 1 - (1 - P(\text{a process is bufferer and receives M}))^n$$

$$= \quad 1 - (1 - C/n(1 - P_{loss}))^n$$

Then, the probability of failure (all bufferers fail to receive a message) P_{fail} is calculated as follows.

$$P_{fail} = 1 - P_{success} = 1 - P_1 - P_2$$

$$= (1 - C/n(1 - P_{loss}))^n - ((1 - C/n)(1 - P_{loss}))^n$$

Assuming P_{loss} is constant, that is independent of n, as n grows, P_{fail} tends to $e^{-C(1 - Ploss)}$. Thus, given the probability of receive omission, the probability of failure can be adjusted by setting C, independent of the size of the membership. P_{fail} gets exponentially smaller as we increase C. However, in many cases P_{loss} is a function of group size. It depends on the size and topology of the underlying network. For example, in a tree-shaped topology, messages have to go through O(logn) links. If we assume that P_{ll} is the individual link loss, then

$$P_{loss} = 1 - (1 - P_{ll})^t$$

where t is the average number of links that a message has to go through. In this case, t grows with O(logn).

Receive omissions are no longer independent from each other. Setting C, in this case, does depend on n.

7.3 Implementation

In this section, we discuss the design of the hash function H, how we integrate our optimization with an epidemic multicast protocol, and how this impacts the buffering requirements of the protocol.

The hash function H has to be both fair and cheap. It has to be fair to provide that the expected number of bufferers for a message is C and also the messages are buffered evenly over the membership. It has to be cheap, since it is calculated each time a message is received. As mentioned before, cryptographic hashes are typically fair, but not cheap. CRC checks are cheap, but not fair. Therefore, we need to design a new function. Our hash function H uses a table of 256 randomly chosen integers, called the shuffle table. The input to H is a string of bytes, and the output is a number between 0 and 1. The algorithm for the hash function is given in figure 7.2.

```
Unsigned integer Hash = 0

For each byte b do

        Hash = Hash XOR shuffle [b XOR least_signif_byte(hash)]

Return Hash/MAX_INTEGER
```

Figure 7.2. Algorithm of the Hash Function

We have integrated our buffering optimization technique to the pbcast protocol. In general, the technique is applicable to any epidemic multicast protocol. In order to do that, we need to modify the protocol. For this optimization, algorithm of modifications to the Pbcast protocol is given in figure 7.3. We call the new protocol as Pbcast-hash. Previously, each member buffers messages that it received until it is known that the message has become stable. If a member receives a retransmission request for a message, it services the retransmission out of its own buffer. With the optimization, a member that receives a request for a message may not have that particular message buffered locally. If so, by utilizing the hash function the member calculates the set of bufferers for the message, and picks one of the bufferers at random. Then, the member sends a retransmission request directly to the bufferer, specifying the message identifier and the destination address. A bufferer, upon receipt of such a request, determines if it has buffered the message. If so, it services the request. If not, it ignores the request.

```
Case: Receipt of PT_PBCAST_REQUEST
{
if msg is in buffer {
        msg.request_counter ++
        // multicast retransmission to group
        if (msg.request_counter >= threshold) {
                send_retrans(groupid, msg.seqno, data message)
                retrans_count ++
                msg.request_counter = 0 }
        else {
                // basic pbcast: unicast retransmission
                if ((my_round_number == msg.round_number) and (retrans_count_ < limit-retrans))
                        send_retrans(msg.source, msg.seqno, data message)
                        retrans_count ++ }
}
else { // if the requested message is not in buffer
        compute Hash for msg
        send_special_request(randomly selected bufferer, msg.id) }
}
New Event:
Case: Receipt of PT_PBCAST_SPECIAL_REQUEST
{
if msg is in buffer {
        msg.request_counter ++
        // multicast retransmission to group
        if (msg.request_counter >= threshold) {
                send_retrans(groupid, msg.seqno, data message)
                retrans_count ++
                msg.request_counter = 0 }
        else {
                if (retrans_count_ < limit-retrans) {
                        send_retrans(actual requestor, msg.seqno, data message)
                        retrans_count ++ } }
}
else ignore
}
```

Figure 7.3. Algorithms for Pbcast-hash

Note that, although members do not buffer all messages they receive, they still have to maintain some information about the messages. In the original protocol, members had to buffer all messages until they are believed to be stable. In the optimized protocol, members only need to keep the identifiers of messages they have received. This can be done in sorted lists of records, one list per sender. Each record describes, using two sequence numbers, a range of consecutively received messages. Since there are typically not many senders, and each list will have a small size consisting of one or a few entries, the amount of storage is negligible.

Our optimization improves the buffering requirements of the epidemic protocol as follows. In the original protocol, the memory requirement for buffering on each member grows $O(\rho*\log n)$ where ρ is the total message rate and n is the number of participants. We assume fixed sized messages and fixed message loss rate (Birman et al., 1999). The number of rounds of gossip required to spread information fully with a certain probability grows $O(\log n)$. In the optimized protocol, the buffering requirement on each member shrinks by $O(\rho*\log n/n)$ since C is constant.

7.4 Improvement

Until now we have assumed that message loss was due to rare and independent receive omissions. In this section, we will suggest an improved strategy in order to deal with more catastrophic message loss, without sacrificing the advantageous scaling properties. The improvement consists of two parts.

1. Maintaining two message buffers

2. Multicast retransmissions

In the first part, we assume two message buffers, namely long-term and short-term. The long-term buffer is the one in which messages are kept by the corresponding bufferers. The short-term buffer keeps all messages in FIFO order as they are received, for some fixed amount of time. Since messages are kept for a fixed amount of time, the size of

short-term buffer is linearly dependent on the message rate ρ, but independent of group size n. Both buffers can be used for retransmissions during message loss recovery.

The second part includes an enhancement to the loss recovery phase of the multicast protocol. We have already proposed and implemented similar strategies for Pbcast. We call these optimized protocols pbcast-ipmc and pbcast-grb. Details of the protocols are found in chapter 5. The idea is to detect send omissions and large correlated receive omission problems, and use multicast rather than unicast for retransmissions.

To support this improvement, a typical epidemic protocol can be modified as follows. Members detect gaps in the multicast message stream by inspecting sequence numbers. They include information about gaps in gossip messages. When a member receives a gossip with information about a gap that it has detected as well, it sends a multicast retransmission request request to the sender. The probability of such an event is low in case of a few receive omissions, but high in case of a catastrophic omission. The sender should still have the message in its short-term buffer to meet the retransmission request. Since these retransmission requests are only triggered by randomized gossip messages, this will not lead to implosion problems seen in ack or nak based protocols.

These improvements, together, lead to two significant benefits. First, they make catastrophic loss unlikely, so that the assumptions of the basic optimization are mostly satisfied. Secondly, since most message loss is detected quickly, retransmissions will be satisfied out of the short-term buffer without the need for retransmission of requests to the bufferers. Then, the long-term buffer is only necessary for all-or-none semantics in rare failure scenarios.

7.5 Simulation study

To validate the buffering mechanisms, we have conducted a simulation study of pbcast with and without our optimization. We have used pbcast-ipmc protocol supporting a multicast retransmission scheme. As the underlying environment, we used the ns-2 network simulator.

7.5.1 Topologies and process groups

In our simulation study, we constructed two different topologies with sizes ranging from 20 to 128 nodes. One of them is a pure tree topology with the sender located at the root node and receivers located at each node that forms a dense session. A sample tree topology containing 100 nodes is illustrated in figure 7.4. The other one is a transit-stub topology with the sender located on a central node and receivers located at each node. We used gt-itm topology generator for producing transit-stub topologies. A sample transit-stub topology with 128 nodes is shown in figure 7.5.

A certain link noise rate is set on every link forming a system-wide noise. We varied three operating parameters, namely group size, multicast message rate of the sender, and system-wide noise rate. We conducted extensive simulations to analyze buffering behavior of protocols and the impact of our optimization on the buffering behavior of an epidemic protocol. Our analysis study mainly focuses on the following cases for each topology and protocol.

- Mean buffer requirement of group members as a function of group size
- Mean buffer requirement of group members as a function of multicast message rate
- Mean buffer requirement of group members as a function of link loss rate
- Buffer requirements of individual group members

For our optimization, we also analyzed the number of bufferers for multicast messages, and the impact of link loss probability on this value

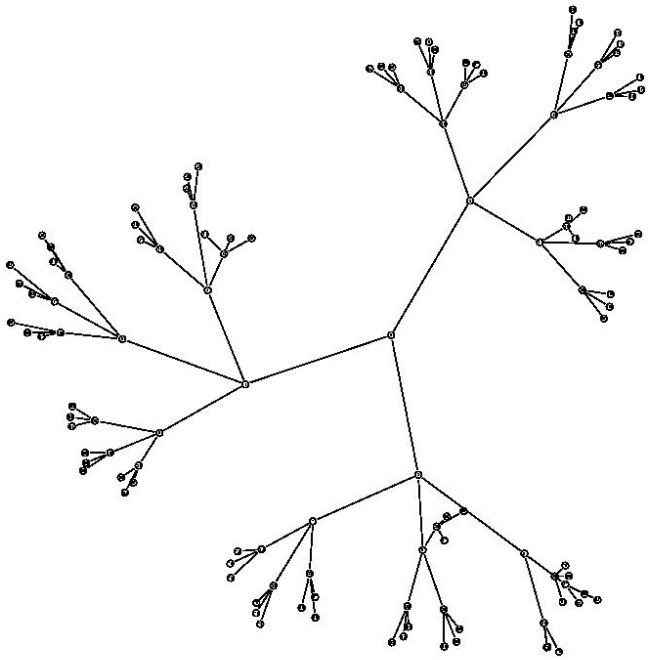


Figure 7.4. A tree topology

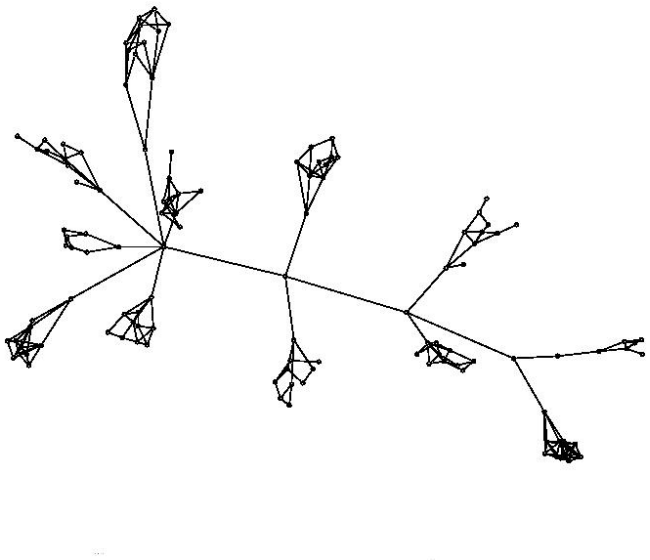


Figure 7.5. A transit-stub topology

7.5.2 Results and Analysis

In our first analysis, our interest lies in the required amount of buffer space of a typical participant, as a function of group size. We measured the maximum number of messages that needed to be buffered at a typical participant. In our simulations, the individual link

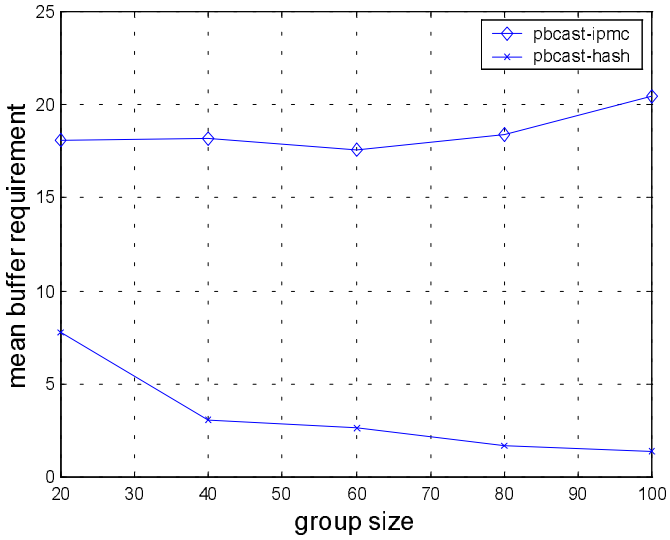
loss probability, P_{ll}, in the network is 0.1%. In these cases, C is approximately 6. We varied the operating parameters of multicast data rate of the sender and group size. Figure 7.6(a) and (b) show results for tree and transit-stub topologies respectively, where the multicast data rate is 25 messages per second. Figure 7.7 shows results for the case where multicast data rate is 100 messages per second. Our findings are as follows. Buffering optimization greatly reduces the memory requirements of group members. Furthermore, the buffering behavior is more predictable for pbcast-hash protocol. In fact, we observed that as the group size scales up, buffering requirement of typical participant decreases.

We also analyzed the effect of multicast message rate and link noise rate on the buffering requirements of pbcast-ipmc and pbcast-hash. Figure 7.8 gives results for multicast message rate versus mean buffer requirement of participants. For these simulations, the message rate is varied from 25 to 100 messages per second. Link loss probability is fixed at 0.1%. We used a tree topology with 100 members. For pbcast-ipmc protocol, we observe a linear relationship between application message rate and mean buffer requirement. Pbcast-hash, on the other hand, reduces the buffer requirement of participants, and shows a very small increase relative to pbcast-ipmc as the application message rate increases.

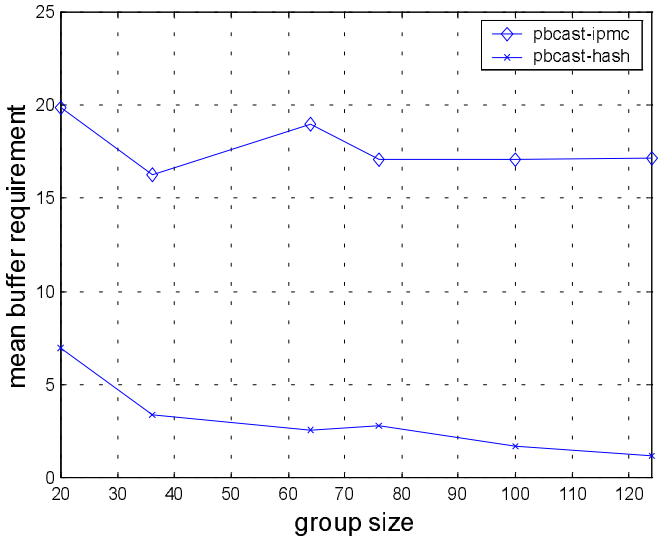
Figure 7.9 illustrates the impact of link loss probability. For these set of simulations, we fixed the message rate at 100 messages per second and varied link loss probability from 0.1% to 1.5%. Pbcast-ipmc protocol shows an increase in the buffer requirement as the system-wide noise rate increases, whereas pbcast-hash has almost constant buffering needs.

Now, we focus on the buffer requirements of individual group members for both protocols. Figure 7.10 shows analysis results for a tree topology simulation with 100 members. For figure 7.10(a) and (b) the message rate of the sender is 25 and 100 messages per second, respectively. Our results validate that the optimization significantly reduces the memory requirements on each individual member, and also that the buffering responsibility is spread evenly over all members.

Additionally, for pbcast-hash protocol, we analyzed the number of bufferers for individual data messages and the impact of the link loss probability on these numbers. In figure 7.11, for 500 consecutive multicast messages starting from message with sequence number 1000, we show on how many locations each of the messages was buffered for two different link loss probabilities: a) 0.1% and b) 1.5%. With large loss rates, in order to get the same P_{fail} probability, it is necessary to buffer messages in more locations. For this reason, the probability that nobody buffers a message ($1-P_2$) is actually smaller for situations with larger loss. Our results demonstrate this effect clearly. Note that, in figure 7.11(a) three messages were not buffered anywhere. This does not imply that the messages were not delivered to every member. In fact, in these simulations, all messages were correctly delivered.

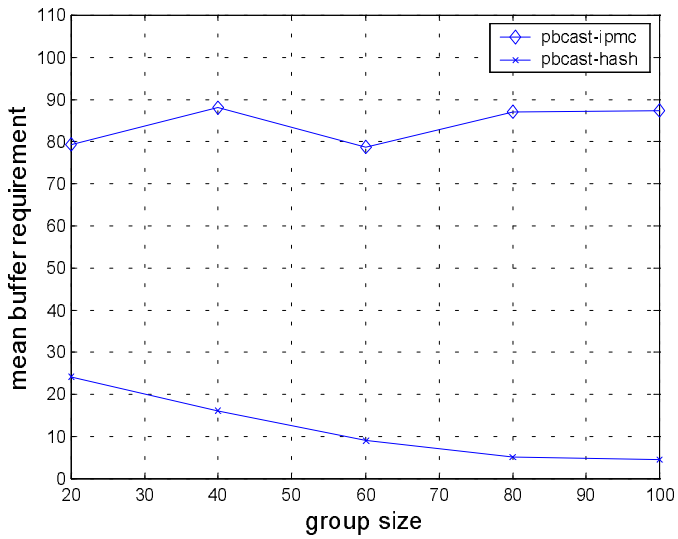


(a)

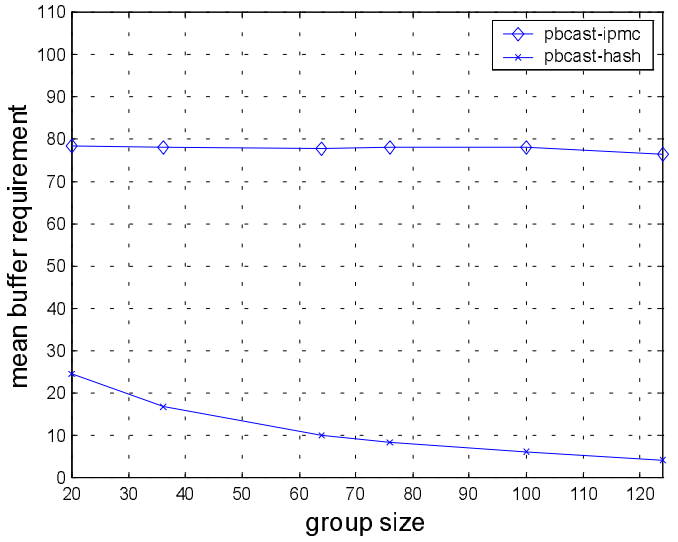


(b)

Figure 7.6. The required amount of buffer space for a typical member as a function of group size. Multicast message rate is 25msgs/sec. a) Tree topology, b) Transit-stub topology



(a)



(b)

Figure 7.7. The required amount of buffer space for a typical member as a function of group size. Multicast message rate is 100msgs/sec. a) Tree topology, b) Transit-stub topology

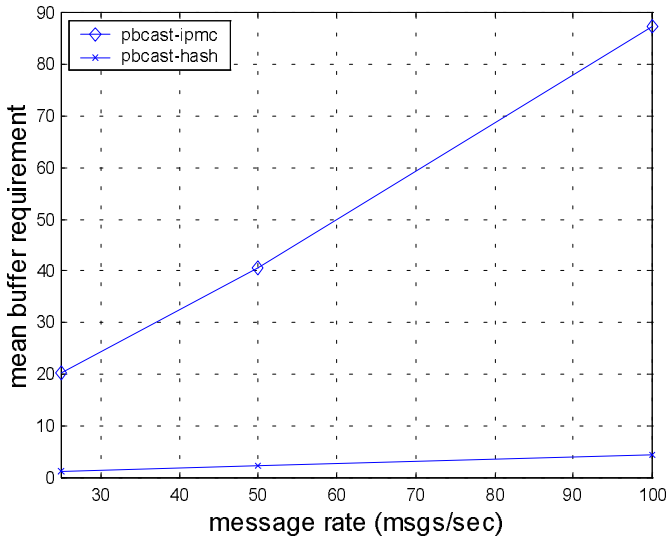


Figure 7.8. Average amount of buffer space per member as a function of message rate

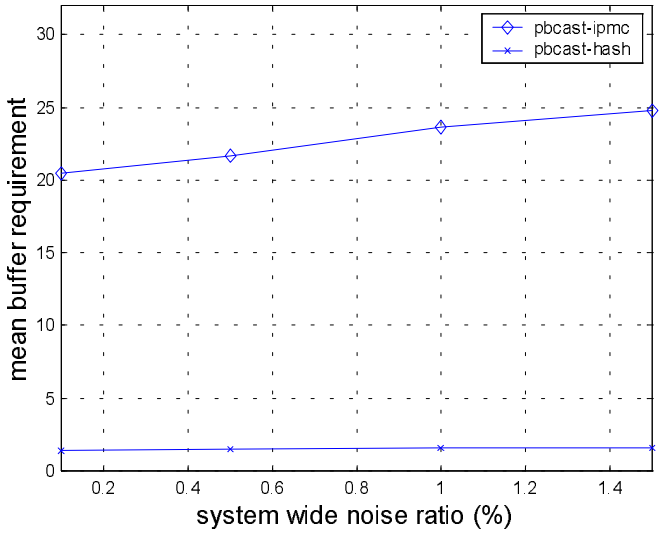
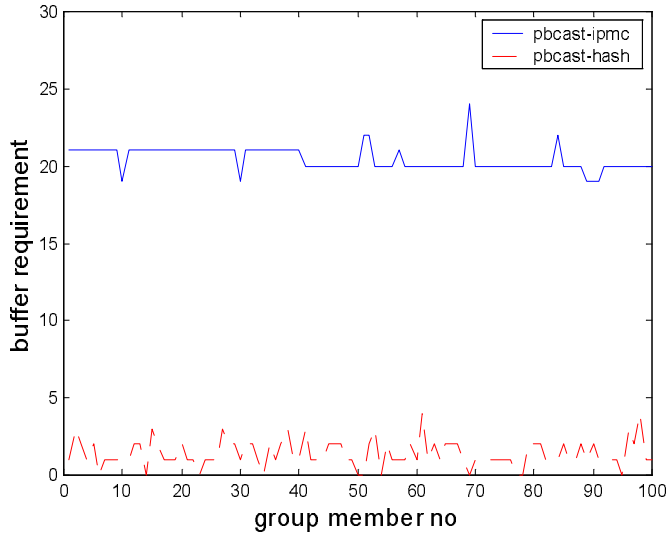
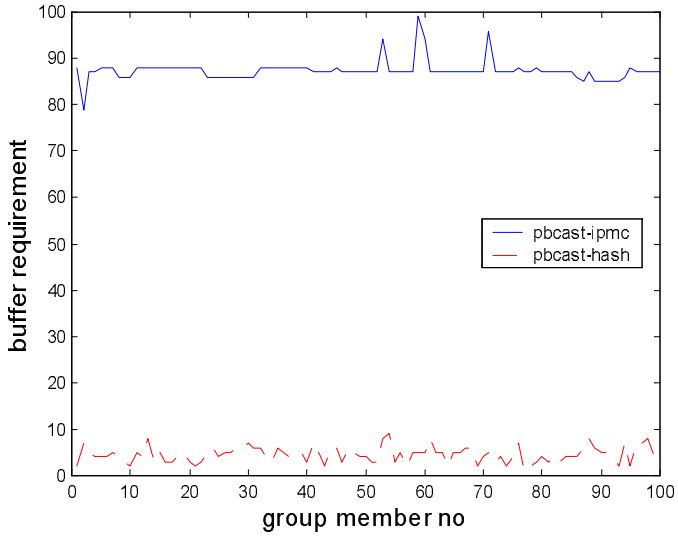


Figure 7.9. Average amount of buffer space per member as a function of link loss probability

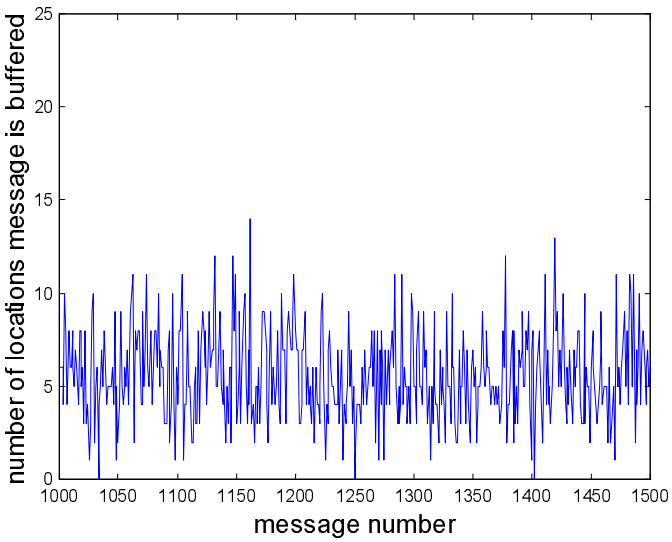


(a)

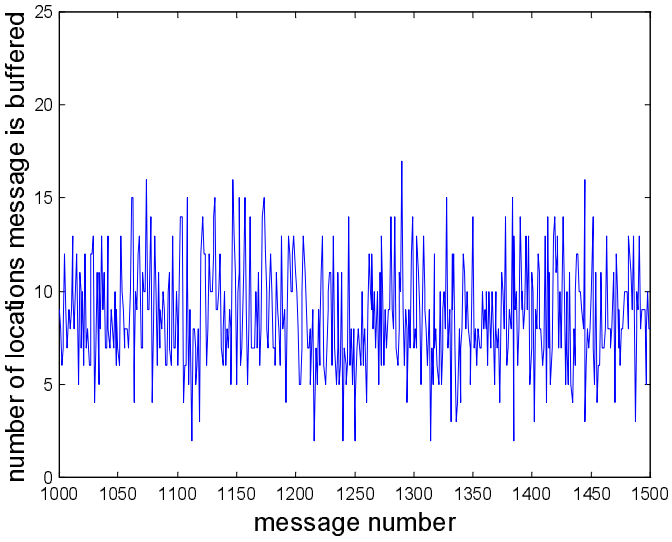


(b)

Figure 7.10. Buffer requirements of individual group members. a) message rate is 25msgs/sec, b) message rate is 100msgs/sec



(a)



(b)

Figure 7.11. Number of locations that individual messages are buffered. a) Link loss probability is 0.1%, b) link loss probability is 1.5%

7.6 Summary

This chapter starts with describing a technique for efficient buffering in reliable multicast protocols. In this study, we conducted a simulation study to validate how the technique significantly optimizes buffer requirements of group participants. The chapter explains the simulation study, results and analysis. We described how the optimization can be incorporated into an epidemic multicast protocol. Based on our simulation model for Pbcast protocol (described in chapter 5), we implemented Pbcast-hash protocol on ns-2 network simulator. Analysis results demonstrate that the technique is highly effective. The buffer requirements on a group member are reduced by a factor of n/C, where n is the size of the group, and C is a small constant containing the number of sites where a message should be buffered.

8. Conclusion

This thesis study investigated the issues of scalability, throughput stability and efficient buffering in reliable multicast protocols. The focus is on the analysis of a new class of scalable reliable multicast protocol, Pbcast, that is based on an epidemic loss recovery mechanism. The protocol offers scalability, throughput stability and a bimodal delivery guarantee. A theoretical analysis study for the protocol is already available (Birman et al., 1999). In this study, we developed experimental and simulation models for the protocol, and conducted extensive analysis studies for investigating protocol properties in practice and comparing it with other classes of reliable multicast protocols across various network characteristics and application scenarios.

General results of our study can be described as follows. In the experimental model, we showed that the throughput behavior of Pbcast remains stable as the process group size is scaled, and the protocol is more stable and scalable compared to the virtually synchronous reliable multicast protocols in several network and application scenarios. The scenarios investigated include soft process failures and system-wide message loss. Our analysis study focuses on the overhead and throughput of the protocols. In contrast to the experimental model, our simulation model enabled us to evaluate protocol performance on several network topologies, failure models, group application scenarios and large scale settings up to thousand nodes. Furthermore, we compared Pbcast with a scalable reliable multicast protocol, SRM, offering best-effort reliability, and developed some optimizations to Pbcast. We showed that, as the network and process group size scale up, the number of control messages received by group members during loss

recovery increases linearly for SRM protocols. In effect, SRM protocols have higher bandwidth consumption compared to Pbcast protocols. We investigated the inter-arrival distributions of data messages for the protocols. We showed that, inter-arrival times of data messages are stable with an increase in the group size for Pbcast, and the distribution changes with the group size, hence it is not stable for SRM. This is mainly due to the higher number of repair messages received by group members during loss recovery and its dependence on group size. Previously, stable throughput is not normally considered to be a critical requirement in reliable multicast protocols, but as discussed in chapter 2, we believe that there are a substantial number of applications for which such a guarantee is important. We also accomplished a detailed study of message latency behavior of Pbcast and SRM protocols. Results show that on large-scale networks, node-level message latencies of Pbcast protocol is smaller compared to SRM's message latencies. We observed that as SRM is scaled to larger groups, steadiness of throughput can be expected to degrade. We analyzed configurations, such as local area networks connected by long distance links and networks where routers with limited bandwidth connect group members, that are common in today's networks. We showed that, high overhead rate can cause routers in a wide area network to become saturated easily. We discussed additional results in detail in Section 6.11. In addition, we presented a technique for buffer optimization in reliable multicast protocols, and conducted a simulation study to validate how the technique significantly optimizes buffer requirements of group participants. Analysis results demonstrate that the technique is highly effective. The buffer requirements on a group member are reduced by a factor of n/C, where n is the size of the group, and C is a small constant containing the number of sites where a message should be buffered.

Inverted protocol stack is a new approach for overcoming throughput instability and scalability problems of traditional reliable multicast protocols. In this thesis study, we demonstrated how this approach works well on several network settings. Under the light of these results, a future work in this area of research would be developing and analyzing effectiveness of the approach in real large-scale networks. In fact, such an attempt is currently under development within the Spinglass[2] project of Cornell University, Department of Computer Science. The project is based on this new approach in which the

[2] http://www.cs.cornell.edu/Info/Projects/Spinglass/index.html

core protocols supporting multicast data transmission give probabilistic reliability guarantees. The project seeks to implement a system around this class of protocols, embedding them into the major software architectures and network operating systems, and to show how applications can be constructed on the resulting probabilistic infrastructure.

An additional area for further study within our simulation model would be a detailed exploration of hierarchical gossip mechanisms for the protocol. The hierarchical gossip approach, which is discussed in Section 3.7, would help to overcome the following two drawbacks of the protocol in terms of scalability. First, each process needs a full membership information for the multicast group, since this information is required by the anti-entropy protocol. But, for large-scale groups, group membership information can become too large and membership updates cause high traffic on the network. Second, in a large network, anti-entropy protocol will involve communication over high-latency paths. Then buffering requirements and round length parameter of the protocol grow as a function of worst-case network latency.

REFERENCES

Babaoglu, O., Davoli, R., Giachini, L. and Baker, M., 1995, Relacs: A Communications Infrastructure for Constructing Reliable Applications in Large-Scale Distributed Systems, Technical Report, UBLCS-94-15, Laboratory for Computer Science, Univ. of Bologna, Italy.

Bailey, N.T.J., 1975, The Mathematical Theory of Infectious Diseases and its Applications, second edition, Hafner Press.

Bajaj, S., Breslau, L., Estrin, D., et al., 1999, Improving Simulation for Network Research, USC Computer Science Dept. Technical Report 99-702.

Baldoni, R., Mostefaoui, A. and Raynal, M., 1996.a, Causal Delivery of Messages with Real-Time Data in Unreliable Networks, *Journal of Real-Time Systems*, 10(3), 245-262p.

Baldoni, R., Prakash, R., Raynal, M. and Singhal, M., 1996.b, Broadcast with Time and Causality Constraints for Multimedia Applications, Technical Report 2976, INRIA (France).

Birman, K.P. and Joseph, T.A., 1987, Exploiting Virtual Synchrony in Distributed Systems, *Proceedings of the 11th Symposium on Operating System Principles*, New York: ACM Press, 123-128p.

Birman, K.P. and van Renesse, R., 1994, Reliable Distributed Computing with the Isis Toolkit, New York: IEEE Computer Society Press.

Birman, K.P., 1993, The Process Group Approach to Reliable Distributed Computing, *Communications of the ACM*, 36(12), 37-53p.

Birman, K.P., 1997, Building Secure and Reliable Network Applications, Manning Publishing Company and Prentice Hall, Greenwich, CT. http://www.browsebooks.com/Birman/index.html

Birman, K.P., 1999, A Review of Experiences with Reliable Multicast, *Software Practice and Experience*.

Birman, K.P., Hayden, M., Ozkasap, O., Xiao, Z., Budiu, M. and Minsky, Y., 1999, Bimodal Multicast, *ACM Transactions on Computer Systems*, 17(2), 41-88p.

Birrell, A.D., Levin, R., Needham, R.M. and Schroeder, M.D., 1982, Grapevine, An Exercise in Distributed Computing, *Communications of the ACM*, 25(4), 260-274p.

Cheriton, D. and Zwaenepoel, W., 1985, Distributed Process Groups in the V Kernel, *ACM Transactions on Computer Systems*, 3(2), 77-107p.

Clark, D. and Tennenhouse, D.L., 1990, Architectural Considerations for a new Generation of Protocols, *Proceedings of the '90 Symposium on Communications Architectures and Protocols*, Philadelphia, PA, 200-208p.

Coulouris, G., Dollimore, J. and Kindberg, T., 1994, Distributed Systems - Concepts and Design, Addison-Wesley Publishing Company, 2nd edition.

Cristian, F., 1996, Synchronous and Asynchronous Group Communication, *Communications of the ACM*, 39(4), 88-97p.

Cristian, F., Aghili, H., Strong, R. and Dolev, D., 1985, Atomic Broadcast: From Simple Message Diffusion to Byzantine Agreement, *Proc. 15th International FTCS*, 200-206p.

Deering, S. and Cheriton, D., 1990, Multicast Routing in Datagram Internetworks and Extended LANs, *ACM Transactions on Computer Systems*, 8(2), 85-110p.

Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D. and Terry, D., 1987, Epidemic Algorithms for Replicated Database Maintenance, *Proceedings of the Sixth ACM Symposium on Principles of Distributed Computing*, Vancouver, British Columbia, 1-12p.

Dolev, D. and Malki, D., 1996, The Transis Approach to High Availability Cluster Communication, *Communications of the ACM*, 39(4), 64-70p.

Floyd, S., Jacobson, V., Liu, C., McCanne, S. and Zhang, L., 1997, A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing, *IEEE/ACM Transactions on Networking*, 5(6), 784-803p. http://www-nrg.ee.lbl.gov/floyd/srm.html

Golding, R.A., Long, D.D. and Wilkes, J., 1994, The Refdbms Distributed Bibliographic Database System, *USENIX Winter 1994 Technical Conference Proceedings*.

Golding, R.A. and Taylor K., 1992, Group Membership in the Epidemic Style, Technical Report, UCSC-CRL-92-13, University of California at Santa Cruz.

Guo, K., 1998, Scalable Message Stability Detection Protocols, Ph.D. dissertation, Cornell University Dept. of Computer Science.

Hadzilacos, V. and Toueg, S., 1993, Fault-tolerant Broadcasts and Related Problems, Distributed Systems, Chapter 5, edited by Mullender, S., ACM Press, 2nd edition, 97-145p.

Hanle, C. and Hofmann, M., 1998, Performance comparison of Reliable Multicast Protocols using the Network Simulator ns-2, *Proceedings of the Annual Conference on Local Computer Networks*, Boston, MA.

Hayden, M., 1998, The Ensemble System, Ph.D. dissertation, Cornell University Dept. of Computer Science.

Hayden, M. and Birman, K.P., 1996, Probabilistic Broadcast, Technical Report, TR96-1606, Department of Computer Science, Cornell University.

Keshav, S., 1988, REAL: A Network Simulator, UCB CS Technical Report 88/472.

Kumar, V., 1995, Mbone: Interactive Multimedia on the Internet, New Riders Publishing, Indianapolis, Indiana, USA.

Labovitz, C., Malan, G.R. and Jahanian, F., 1997, Internet Routing Instability, *Proceedings of SIGCOMM `97*, 115-126p.

Ladin, R., Lishov, B., Shrira, L. and Ghemawat, S., 1992, Providing Availability using Lazy Replication, *ACM Transactions on Computer Systems*, 10(4), 360-391p.

Lamport, L., 1978, The Implementation of Reliable Distributed Multiprocess Systems, *Computer Networks*, 2, 95-114p.

Li, D. and Cheriton, D.R., 1998, OTERS (On-Tree Efficient Recovery using Subcasting): A Reliable Multicast Protocol, *Proceedings of the 6th IEEE International Conference on Network Protocols (ICNP'98)*, 237-245p.

Lidl, K., Osborne, J. and Malcome, J., 1994, Drinking from the Firehose: Multicast USENET News, *USENIX Winter 1994*, 33-45p.

Lin, J.C. and Paul, S., 1996, A Reliable Multicast Transport Protocol, *Proceedings of IEEE INFOCOM '96*, 1414-1424p. http://www.bell-labs.com/user/sanjoy/rmtp.ps

Liu, C., 1997, Error Recovery in Scalable Reliable Multicast, Ph.D. dissertation, University of Southern California.

Lucas, M., 1998, Efficient Data Distribution in Large-Scale Multicast Networks, Ph.D. dissertation, Dept. of Computer Science, University of Virginia.

Malki, D., Multicast Communication for High Availability, 1994, Ph.D. Thesis, Hebrew Univ. in Jerusalem.

Mishra, S. and Kuntur, S.M., 1999, Improving Performance of Atomic Broadcast Protocols using Newsmonger Technique, *Proceedings of the 7th IFIP International Working Conference on Dependable Computing for Critical Applications*, San Jose, CA, 157-176p.

Mishra, S. and Wu, L., 1998, An Evaluation of Flow Control in Group Communication, *IEEE/ACM Transactions on Networking*, 6(5).

Moser, L.E., Melliar-Smith, P.M., Agarwal, D.A., Budhia, R.K., et.al, 1996, Totem: A Fault-tolerant Multicast Group Communication System, *Communications of the ACM*, 39(4), 54-63p.

Mullender, S., editor, 1993, Distributed Systems, ACM Press, Addison-Wesley Publishing Company, 2nd edition.

Ozkasap, O., Xiao, Z. and Birman, K.P., 1999.a, Scalability of Two Reliable Multicast Protocols, Technical Report, TR99-1748, Department of Computer Science, Cornell University.

Ozkasap, O., Van Renesse, R., Birman, K.P. and Xiao, Z., 1999.b, Efficient Buffering in Reliable Multicast Protocols, *Proceedings of Networked Group Communication Symposium (NGC99)*, Lecture Notes in Computer Science, Springer, Pisa, Italy.

Paul, S., Sabnani, K., Lin, J. C. and Bhattacharyya, S., 1997, Reliable Multicast Transport Protocol (RMTP), *IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multipoint Communication*, 15(3), http://www.bell-labs.com/user/sanjoy/rmtp2.ps

Paxson, V., 1997, End-to-End Internet Packet Dynamics, *Proceedings of SIGCOMM `97*, 139-154p.

Petersen, K., Spreitzer, M.J., Terry, D.B., Theimer, M.M. and Demers, A.J., 1997, Flexible Update Propagation for Weakly Consistent Replication, *Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles*, Saint-Malo, France, 288-301p.

Phidias, http://www.stna.dgac.fr/projects/, http://aigle.stna7.stna.dgac.fr/

Piantoni, R. and Stancescu, C., 1997, Implementing the Swiss Exchange Trading System, *FTCS 27*, Seattle, WA, 309-313p.

Reiter, M., 1996, Distributing Trust with the Rampart Toolkit, *Communications of the ACM*, 39(4), 71-74p.

Speakman, T., Farinacci, D., Lin, S. and Tweedly, A., 1998, PGM Reliable Transport Protocol, Internet-Draft.

Van Renesse, R. and Birman, K.P., 1995, Protocol Composition in Horus, Technical Report, TR95-1505, Department of Computer Science, Cornell University.

Van Renesse, R., Birman, K.P. and Maffeis, S., 1996, Horus: A Flexible Group Communication System, *Communications of the ACM*, 39(4), 76-83p.

Van Renesse, R., Hickey, T. and Birman, K.P., 1994, Design and Performance of Horus: A Lightweight Group Communications System, Technical Report, TR94-1442, Department of Computer Science, Cornell University.

Van Renesse, R., Minsky, Y. and Hayden, M., 1998, A Gossip-style Failure Detection Service, *Proceedings of Middleware'98*, 55-70p.

XTP Forum, 1995, Xpress Transfer Protocol Specification, XTP Rev 4.0.