Available online at www.sciencedirect.com
SCIENCE DIRECT®
Performance Evaluation 57 (2004) 177–198
PERFORMANCE EVALUATION
An International Journal
www.elsevier.com/locate/peva

Performance study of a probabilistic multicast transport protocol

Öznur Özkasap*

Department of Computer Engineering, Koc University, 34450 Istanbul, Turkey

Received 19 March 2001; received in revised form 13 October 2003

Abstract

Traditional reliable multicast protocols depend on assumptions about flow control and reliability mechanisms, and they suffer from a kind of interference between these mechanisms. This in turn affects the overall performance, throughput and scalability of group applications utilizing these protocols. However, there exists a substantial class of distributed applications for which the throughput stability and scalability guarantees are indispensable. Bimodal Multicast (Pbcast) is a new option in scalable reliable multicast protocols that uses an inverted protocol stack approach, in which probabilistic mechanisms are used at low layers, and reliability properties introduced closer to the application. The main contributions of this study are development of simulation models for performance evaluation of Bimodal Multicast, demonstration of how the inverted protocol stack approach works well on several network settings, and its comparison with best-effort reliable multicast mechanisms. Analysis results reveal that Bimodal Multicast, together with optimizations for improving its latency and reliability characteristics, scales well, exhibits stable throughput and in contrast to the other scalable reliable multicast mechanisms it gives predictable reliability even under highly perturbed conditions.

Keywords: Bimodal multicast (Pbcast); Multicast loss recovery; Scalable group communications; Protocol performance evaluation; Scalable reliable multicast (SRM)

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 [6] and the Mbone [14] 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.

* Tel.: +90-212-338-1548; fax: +90-212-338-1584.
E-mail address: oozkasap@ku.edu.tr (Ö. Özkasap).

0166-5316/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.peva.2003.10.005

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. The degree of reliability guarantees required by distributed 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.

In the first class, there is a great deal of work on communication tools offering reliable multicast protocols for distributed applications [4]. Example systems include Isis [3], Horus [28], Totem [20], Transis [8], Relacs [1] and Ensemble [12]. 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 [26] the maximum number of participants must not exceed about 50–100. Otherwise, transient performance problems can cause these protocols to exhibit degraded throughput.

The second class of protocols focuses on best-effort reliability in large-scale systems. *Best-effort reliability* indicates that a multicast message is not guaranteed to arrive intact to all members of the group or in the same order relative to the other messages. 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. Example systems are Internet Muse protocol for network news distribution [17], the Scalable Reliable Multicast (SRM) protocol [9], the Pragmatic General Multicast (PGM) protocol [10], and the Reliable Message Transfer Protocol (RMTP) [24].

Two approaches that are representatives of many existing solutions for providing loss recovery in scalable multicasting are *nonhierarchical feedback control and hierarchical feedback control*. Overall, the key issue is to reduce the number of feedback messages that are returned to the sender. In the former approach, a model that has been adopted by several wide-area applications is referred to as feedback suppression. An improvement to enhance scalability is referred to as local recovery, which is related to restraining the recovery of a message loss to the region where the loss has occurred. In the latter approach, hierarchical approaches are adopted for achieving scalability for very large groups of receivers [27]. Another alternative for ensuring reliability is forward error correction (FEC). The idea behind this approach is predicting losses and transmitting redundant data.

SRM is a well-known reliable multicast protocol based on feedback suppression. When a receiver detects that it has missed a message, it multicasts its feedback to the rest of the group. Multicasting feedback allows another group member to suppress its own feedback. A receiver lacking a message schedules a feedback with some random delay. SRM 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. ALF principle 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. In contrast

to Bimodal Multicast providing a form of reliability that can be rigorously quantified, SRM provides best-effort reliability.

PGM is a reliable multicast protocol utilizing FEC together with a hierarchical approach and NAK suppression. It offers ordered, duplicate-free multicast data delivery, and 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 an NAK-based loss recovery mechanism and runs over a datagram multicast protocol such as IP multicast.

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.

These protocols are 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 [15,25].

Within the spectrum of scalable reliable multicast protocols, Bimodal Multicast, or Pbcast (probabilistic multicast) in short [5] is a novel option based on the inverted protocol stack approach. The behavior of the protocol can be predicted given simple information on how processes and the network behave most of the time. The protocol scales well and provides predictable reliability with a steady throughput, even under highly perturbed network conditions. In contrast, this kind of behavior can cause other reliable multicast protocols to exhibit unstable throughput. The main contributions of this study are development of simulation models for performance evaluation of Bimodal Multicast, demonstration of how the inverted protocol stack approach works well on several network settings, and its comparison with best-effort reliable multicast mechanisms of SRM. Analysis results reveal that Bimodal Multicast, together with optimizations for improving its latency and reliability characteristics, scales well, exhibits stable throughput and in contrast to the other scalable reliable multicast mechanisms it gives predictable reliability even under highly perturbed conditions.

The present article is organized as follows. Section 2 explains the inverted protocol stack approach and related work. In Section 3, Bimodal Multicast protocol, its implementation within this study, and the differences between the protocol and SRM are described. Section 4 focuses on optimizations to Bimodal Multicast that are developed in this study, for fast loss recovery and better reliability. In Section 5, our analysis results are given followed by an overall discussion in Section 6. Section 7 states conclusions.

2. Inverted protocol stack and related work

Traditional transport protocols utilizing flow control and deterministic reliability mechanisms at lower layers near the network combat with low-probability random events that can restrict the scalability of higher-level reliability abstractions. The lowest layers of a typical protocol stack assume steady and predictable behavior. As the system scales in number of participants and geographical size, the likelihood

that the system will experience low-probability events such as link noise, network scheduling delays, and transient process failures rises causing performance degradation.

One approach to overcome these problems is to construct large-scale reliable protocols using an inverted protocol stack. The method is based on the idea that the lower levels of the protocol stack offer probabilistic guarantees by utilizing randomized mechanisms. Reliability and other strong properties such as message ordering and security are introduced closer to the application, in other words moved to the upper layers. The argument is that the approach as adopted by Bimodal Multicast supports desired application-level semantics and provides scalability as well. In Bimodal Multicast, the lowest layers provide probabilistic data dissemination via epidemic paradigm. The flow control mechanisms are based on constraining data rates. The application-level stronger reliability mechanisms such as virtual synchrony and security build on these probabilistic abstractions. A key property is that lower layers run in a predictable manner since epidemic mechanisms have stable cost independent of system size. A review on the scalability problem for multicast protocols, and how epidemic techniques can be used to solve the problem is available in [29] for the interested reader. In this article, we focus on performance analysis of Bimodal Multicast, its optimized versions, and demonstrate how the inverted protocol stack approach works well on several network settings.

Along these lines, recent work of Gupta et al. [11] examines scalability of strong multicast reliability properties in distributed systems. As argued, traditional protocol designs often disregard the high cost of infrequent events. The frequency and the overall cost of such events often increase as the distributed system scales. Gupta et al. [11] constructs an inverted protocol stack that implements virtually synchronous multicast, and demonstrates the scalability of the protocol over traditional implementations.

One of our related experimental studies has been performed on the SP2 system of Cornell Theory Center that offers an isolated network behavior [23]. In this work, multicast protocols of Ensemble group communication system [12] offering strong reliability guarantees have been compared to Bimodal Multicast. We have focused on the Ensemble multicast protocols in the case of soft process failures, and showed that even a single perturbed group member impacts the throughput of unperturbed members negatively. On the other hand, Bimodal Multicast achieves the ideal throughput rate even with high percentage of perturbed members, and the throughput behavior of the protocol remains stable as the process group size scales up. Furthermore, our results confirm that overhead on the correct processes is bounded as the size of process group increases. Detailed information on the experimental results is given in [21,23].

3. Bimodal multicast protocol

Bimodal Multicast [5] is a novel option in the spectrum of multicast protocols that is inspired by prior work on epidemic protocols [7], Muse protocol for network news distribution [17], and the lazy transactional replication method of [16]. Bimodal Multicast is based on an epidemic loss recovery mechanism. It has been shown to exhibit stable throughput under failure scenarios that are common on real large-scale networks [5]. In contrast, this kind of behavior can cause other reliable multicast protocols to exhibit unstable throughput. Bimodal Multicast consists of two sub-protocols, namely an optimistic dissemination protocol and a two-phase anti-entropy protocol which are described next.

Optimistic dissemination: This sub-protocol 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, the protocol model implemented on ns-2 network simulator [2] in this study uses IP multicast. Otherwise, a randomized dissemination protocol can play this role.

Two-phase anti-entropy: The second stage of Bimodal Multicast 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. The two-phase anti-entropy protocol progresses through unsynchronized gossip rounds. In each round:

- Every group member selects another group member at random and sends a digest of its current message buffer contents. The digest just includes the identifiers of messages in the buffer. The message including the digest is called a 'gossip message'.
- The receiving group member compares the digest with its own message buffer contents. Then, if it is lacking a message, it requests the message from the gossiping process. This message is called 'solicitation', or retransmission request.
- Upon receiving the solicitation, the gossiping process retransmits the requested message to the process sending this request.

3.1. Protocol execution

Fig. 1 illustrates the execution of Bimodal Multicast, where 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 optimistic dissemination protocol. Because of

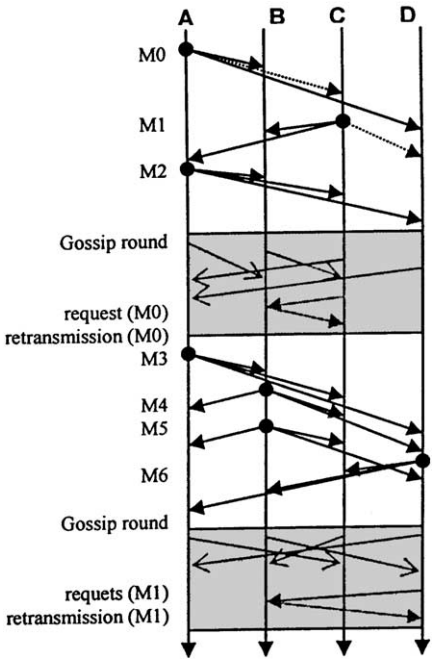

Fig. 1. Execution of Bimodal Multicast.

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 in which each process selects another one at random, and sends its message digest via a gossip message. Upon receiving the 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 by showing that the protocol alternates between dissemination and anti-entropy stages. But, in practice, these stages run concurrently.

One of the differences of Bimodal Multicast'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, the protocol 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. 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 simulations conducted in this study use a round duration of 100 ms. 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. 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.

3.2. Implementation

Bimodal Multicast protocol design on ns-2 consists of three modules as shown in the block diagram of Fig. 2. The first one is the module that performs unreliable data dissemination and uses IP multicast protocol. The second module is the gossip based anti-entropy protocol. The third module accomplishes FIFO message ordering. Our design follows an event-based approach. There are four message types for data, gossip, request and retransmission messages. Protocol messages contain the fields as shown in Fig. 3 where the first entry is the type of the corresponding message.

Every group 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.

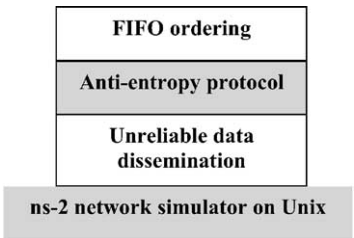

Fig. 2. Bimodal Multicast stack on ns-2.

Message	Fields
Data	<PT_PBCAST_DATA, sequence number, data>
Gossip	<PT_PBCAST_GOSSIP, message buffer digest, round number>
Request	<PT_PBCAST_REQUEST, sequence no of the requested message, round number>
Retransmission	<PT_PBCAST_RETRANS, sequence number of the message retransmitted, data>

Fig. 3. Message fields for the protocol implementation.

Bimodal Multicast protocol agent has the following operating parameters:

- *sub-gsize*: number of members to gossip in each round.
- *step-interval*: gossip round duration. Default is 100 ms.
- *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.

Algorithm for the protocol of our simulation model is given in Fig. 4. The protocol agent runs at every member of a process group application communicating via Bimodal Multicast. In the algorithm, msg denotes a message received by a member. Basically, we define the following four events that trigger the protocol actions:

1. *Receipt of Pbcast data or retransmission message*: When a member receives a data or a retransmission message, the message is buffered, and messages are delivered to the application layer in FIFO order. Also, if some messages are declared as lost, the application is informed about them.
2. *Receipt of Pbcast request message*: 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.

```
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)
```

Fig. 4. Algorithm for Bimodal Multicast protocol on ns-2.

3. *Receipt of Pbcast gossip message*: 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.
4. *Timer interrupt for gossip round*: 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 members (defined by *sub-gsize* parameter) selected at random, and schedules the timer to step-interval value for the next gossip round.

3.3. Comparison of Bimodal Multicast and SRM

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. As well as the reception state, the session messages also contain timestamps that are used to estimate the distance from each member to every other. Members utilize 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 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. Repair requests and retransmissions are multicast to the whole group. A lost packet ideally triggers only a single request from a host just downstream of the point of failure.

The anti-entropy protocol is the part of Bimodal Multicast that deals with loss recovery. During this phase, each process chooses another process in the multicast group at random, 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 a gossip message, the receiving process compares the digest included 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.

Additional message traffic required for loss recovery of both protocols can be explained as follows. We assume that Bimodal Multicast's round duration for gossip is 100 ms, and N is the number of members in the process group. Then, if every process gossips to another process every 100 ms, on the average $N \times 10$ destinations will receive gossip messages every second. Periodic session messages of SRM are transmitted every second in multicast mode. This means that, $N \times N$ destinations will receive session messages every second, and each process receives N session messages every second.

In Bimodal Multicast, 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. For 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 Bimodal Multicast's background overhead is scalable and does not increase with the group size.

4. Optimizations to Bimodal Multicast

In this study, we choose ns-2 as the simulation environment for developing our protocol models. ns-2 is a discrete event simulator for networking research [2] that began as a variant of the REAL network simulator [13]. It provides support for various networking concepts such as routing, multicast protocols (e.g. IP multicast and SRM), link error models, topology and traffic generation. Within our simulation model, we implement Bimodal Multicast protocol framework as described in Section 3.2. Based on the results of analysis studies, for improving latency characteristics and reliability properties of Bimodal Multicast, we develop optimizations to the protocol. In this section, simulation models for these optimizations are described.

4.1. Pbcast-ipmc

Bimodal Multicast uses point-to-point (unicast) communication when retransmitting a message. However, if a message is requested more than once, it is likely that this message loss affects more than one member of the multicast group. Thus, it is appropriate to use multicast communication for retransmission in such a scenario. Pbcast-ipmc is the optimized protocol that exploits multicasting during loss recovery based on a threshold value. Our results show that it leads to fast error recovery and better reliability than Bimodal Multicast under the same network conditions.

At first glance, Pbcast-ipmc has the following advantages. It decreases the request message traffic compared to Bimodal Multicast especially when message losses affect more than one member in the group. Since the optimization exploits multicasting during loss recovery, it increases reliability of the protocol where there exists random noise on the links. Here, there is a trade-off in using multicasting for loss recovery versus resource consumption. Multicasting retransmission leads to faster loss recovery, however, it would consume more resources than unicast recovery especially if the loss event has occurred within one or few group members. That is why, it would not be efficient to use multicast retransmission for all message loss recoveries.

Fig. 5 shows modifications to Bimodal Multicast model needed for implementing Pbcast-ipmc. As it is seen in the algorithm, modification is only needed for the event of request message receipt. For this purpose, 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. If a member receives a retransmission request

```
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) {
            multicast_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 ++ }
```

Fig. 5. Algorithm of modifications for Pbcast-ipmc protocol.

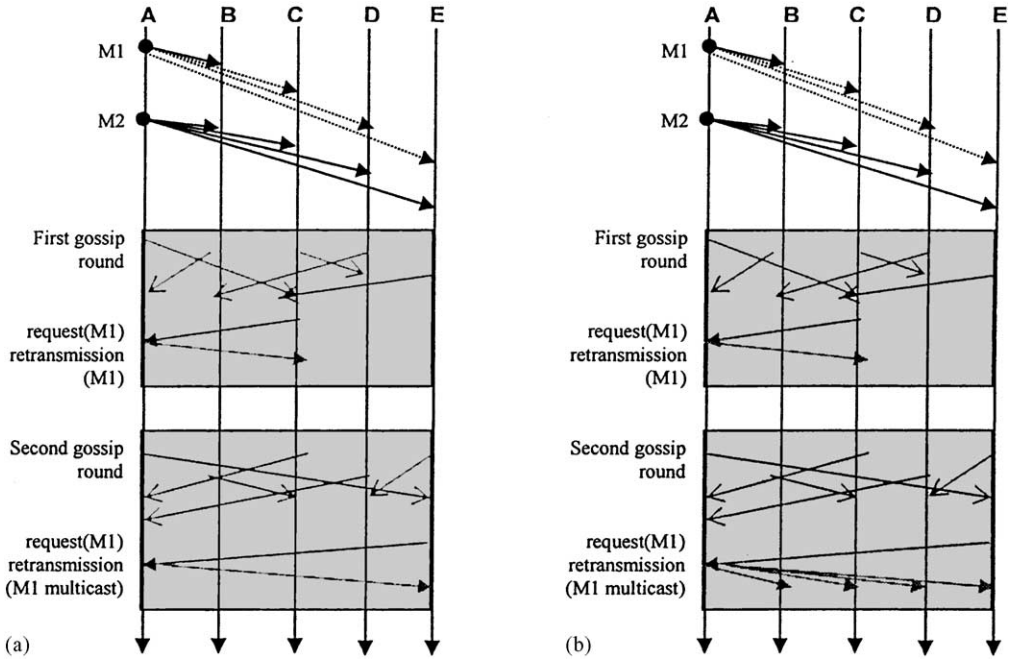

Fig. 6. Sample scenario. (a) Bimodal Multicast. (b) Pbcast-ipmc.

for a message in its buffer, then its request counter is incremented by one. When sending a retransmission for a multicast message, its request counter is first checked. If it exceeds a certain threshold, then the retransmission is multicast to the group. In our implementation, we set the threshold value to 2.

The following scenario, as illustrated in Fig. 6, describes how Pbcast-ipmc performs better than Bimodal Multicast in certain network conditions. Assume that there is a 5-member process group with participants 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 chooses another member at random 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, 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 Bimodal Multicast 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 Bimodal Multicast, 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 Bimodal Multicast, 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 Bimodal Multicast and Pbcast-ipmc illustrate that Pbcast-ipmc increases probability of rapid convergence during loss recovery.

4.2. Pbcast-grb

Pbcast-grb stands for Pbcast with gossip retransmit bit. The optimization consists of Pbcast-ipmc together with the idea of gossip retransmit bit. The idea is to keep information about whether a message is retransmitted previously or not, and include it within gossip messages. Based on this information, members either use multicast or unicast for retransmission. The motivation is that, Pbcast-grb would have benefits over basic Bimodal Multicast in terms of fast and easy loss recovery. It basically utilizes multicasting for some retransmissions based on the retransmit bit information. Pbcast-grb is implemented on ns-2, and analysis results show that similar to Pbcast-ipmc it leads to fast error recovery and better reliability than basic Bimodal Multicast under the same network conditions.

Fig. 7 gives the modifications needed for the implementation of Pbcast-grb protocol within our simulation model. Group members keep a retransmit bit for every message in their buffer. If a member retransmits

```
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 }
```

Fig. 7. Algorithm of modifications for Pbcast-grb protocol.

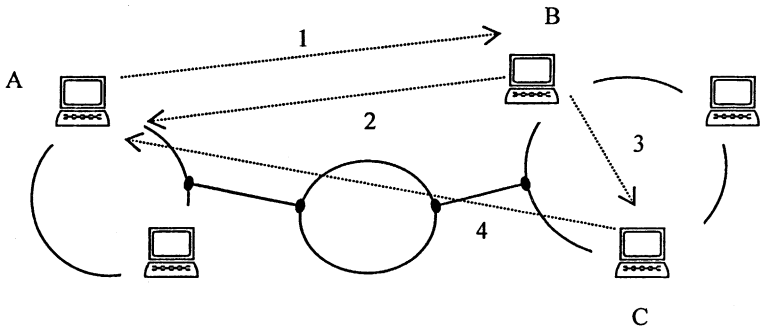

Fig. 8. An illustration of Pbcast-local protocol run.

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 identifier in their message digest. When a member receives a gossip message, it performs 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.

4.3. Pbcast-local

Pbcast-local attempts to perform local loss recovery. It uses neighborhood information among group members and works as illustrated in Fig. 8. Assume that 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 at random, 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 did not multicast message M, it uses multicast to retransmit M.

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 described in step (3). *Mcast* is the special request message sent from neighbor process to gossip sender, as described in step (4). These messages contain the following fields:

Message	Fields
Info	*<PT_PBCAST_INFO, process id, sequence number of the requested message>*
Mcast	*<PT_PBCAST_MCAST, sequence number of the requested message>*

Fig. 9 gives the modifications needed for Pbcast-local protocol. Two new events are defined that are related to receipt of info and mcast messages. Pbcast-local is implemented on ns-2 and our analysis results show that it improves message latency distribution of Bimodal Multicast.

```
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 at random
            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 }}
```

Fig. 9. Algorithm of modifications for Pbcast-local protocol.

5. Performance analysis results

In this section, we include results of our simulation study. The analysis work employs performance metrics that we believe are important when investigating the behavior of scalable reliable multicast protocols. We performed simulations of Bimodal Multicast, optimizations to it for efficient loss recovery, and SRM protocols. In the simulations of protocols on several network topologies and scenarios, operating parameters such as network size, group size, link error rates and multicast data rates are varied. Performance metrics of protocol overhead, throughput, link utilization, inter-arrival distribution, latency distribution and multicast message congestion are analyzed. Additional results are available in [21] for the interested reader.

5.1. Multicast message inter-arrival distributions

This section investigates the inter-arrival distributions of data messages for Bimodal Multicast and SRM, and also the effect of scaling the group size up. For this set of simulations, dense-mode tree-topology networks with sizes ranging from 20 to 60 nodes are constructed. All of the trees have depth 4, and all nodes have a Bimodal Multicast or SRM protocol agent attached. The size of the process group in these simulations equals the network size. There is one sender in the group, it is located at the root node of the tree, and a constant bit rate (CBR) source is attached to the sender which generates 100 messages per second, and the message size is 210 bytes. We configure network links to have bandwidth of 1.5 Mb each. A low-level system-wide constant noise rate is imposed on the network: each link drops 1% of packets passing over it. This loss rate applies to all messages, namely data and control. If a message passes through more than one link to reach its destination, the drop probability would accumulate accordingly, since the same noise rate is set on all links.

Message dissemination rate of the sender, employing a CBR source, is 100 messages per second. Therefore, if no message loss occurs, the inter-arrival time between messages is expected to be 0.01 s. When link noise is introduced to the network, it would result in message drops. Then, loss recovery mechanisms of protocols will generate retransmissions to achieve communication reliability.

Fig. 10(a) shows inter-arrival distributions at a typical receiver for Bimodal Multicast on 20-, 40- and 60-node tree topologies. It is observed that inter-arrival times of data messages are stable as group size scales up. Similarly, Fig. 10(b) shows inter-arrival distributions for SRM. The distribution for SRM changes with the group size hence it is 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 there are a substantial number of applications such as Internet media distribution, electronic stock exchange and distribution of radar and flight track data in air traffic control systems, for which the throughput stability

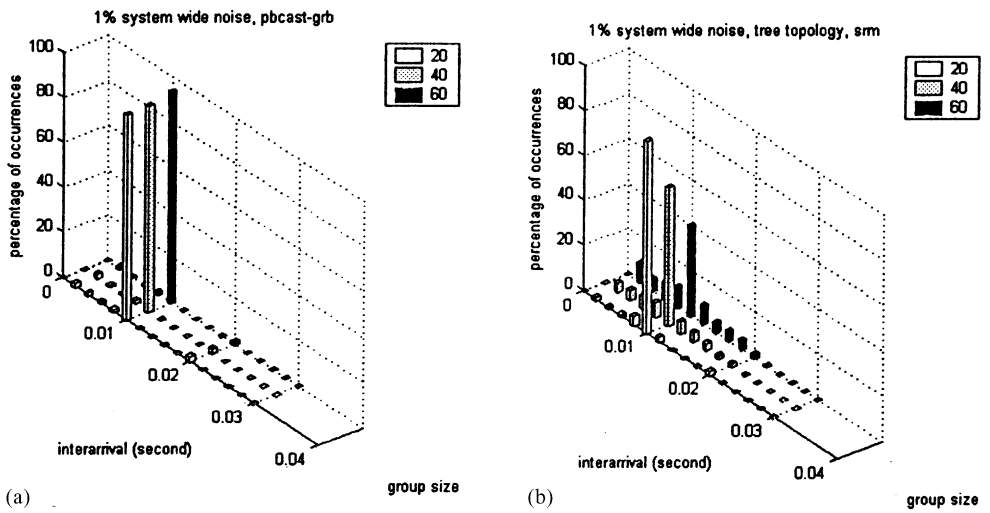

Fig. 10. Histograms of inter-arrival times of Pbcast and SRM with 1% system-wide noise in densely populated tree networks.

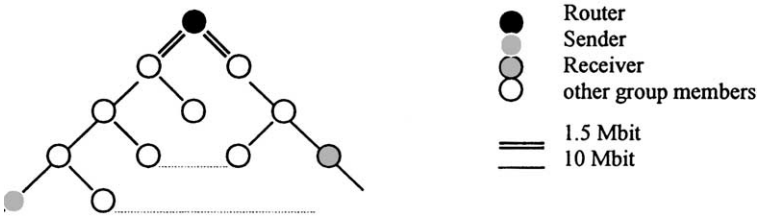

Fig. 11. A tree topology with a router with limited bandwidth at root.

guarantee is extremely important. This property entails the steady delivery of multicast data stream to correct destinations.

5.2. *Impact of limited bandwidth on routers: background overhead and latency distribution*

First set of simulations has focused on the impact of randomized message loss on the performance of the 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. In order to analyze the effect of limited bandwidth on a router, tree topology simulations running on 20-, 40-, 60-, 80- and 100-node networks are constructed. Fig. 11 illustrates such a network where the root node behaves as a router, and links to the router have limited bandwidth of 1.5 Mb compared to the other links of the network that all have bandwidth of 10 Mb. System-wide constant noise rate is set to 1%. As shown in the figure, one of the nodes on the left sub-tree is the sender that sends 100 multicast messages per second, and the message size is 1000 B. Therefore, the sender disseminates 800 Kb 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 analyze the background overhead and latency distribution of the particular receiver on the right sub-tree that is illustrated in the figure.

Fig. 12(a) and (b) shows the background overhead analysis, in the form of request and repair message traffic, respectively. In particular, a dramatic increase in request message traffic of SRM is observed for group sizes larger than 60 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 Bimodal Multicast remains nearly constant, and the growth in repairs is consistent with the size of the group and the system-wide noise rate used in this scenario.

For 20- and 40-node network simulations, message latency distributions of Bimodal Multicast and SRM receiver resemble each other. But, as network and group size scales up, 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. Analysis results for 100-node topology are given in Fig. 13 where the effect of limited bandwidth on a router for SRM protocol can be seen clearly. For comparison, graphs on the left show latency for the particular protocol on the same *x*-axis scale. On the right-hand side, we focus on the detailed latency data of each protocol. Fig. 13(a) and (b) shows the latency distribution of Bimodal Multicast at node level and after FIFO ordering, respectively, for the particular receiver. As it is observed in Fig. 13(c), for the SRM case, a large percentage of messages are delivered with high latency values going up to 15 s.

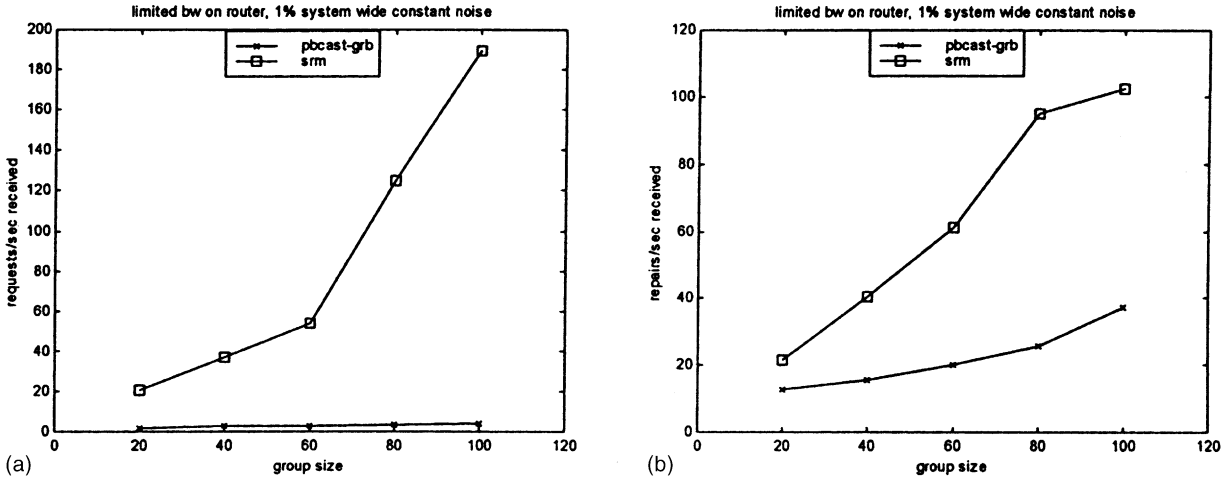

Fig. 12. (a) Requests and (b) repairs in tree topologies with limited bandwidth on root links.

5.3. Impact of Pbcast-local on application level latencies

As mentioned in Section 4.3, Pbcast-local aims to perform local loss recovery. The main benefit of the approach would be decreasing message latencies experienced by a typical group member. In this set of simulations, a 60-node tree topology network is constructed, and 10% constant noise is injected on one outgoing link from the sender, where the sender is located at the root node. All the other nodes are receivers. A receiver that is exposed to some message losses due to the noise on the network link is selected. Then, latency distributions of the receiver for Pbcast-grb and Pbcast-local after FIFO ordering are analyzed.

Fig. 14 shows the latency distributions where Pbcast-local has a notable improvement on the latency characteristics of Bimodal Multicast. Pbcast-local uses neighborhood information among group members and attempts to accomplish local recovery. As our analysis results indicate, the optimization improves latency of data messages.

5.4. Conditions causing multicast message congestion

This section investigates the conditions that cause multicast messages to begin congest when using SRM and Pbcast-grb. The network topology is a 2-cluster, 80-node network where 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 5 ms, while the link delay between clusters is 30 ms. There is 1% 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. Two operating parameters, namely multicast message rate of the sender and inter-cluster noise probability, are varied. 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.

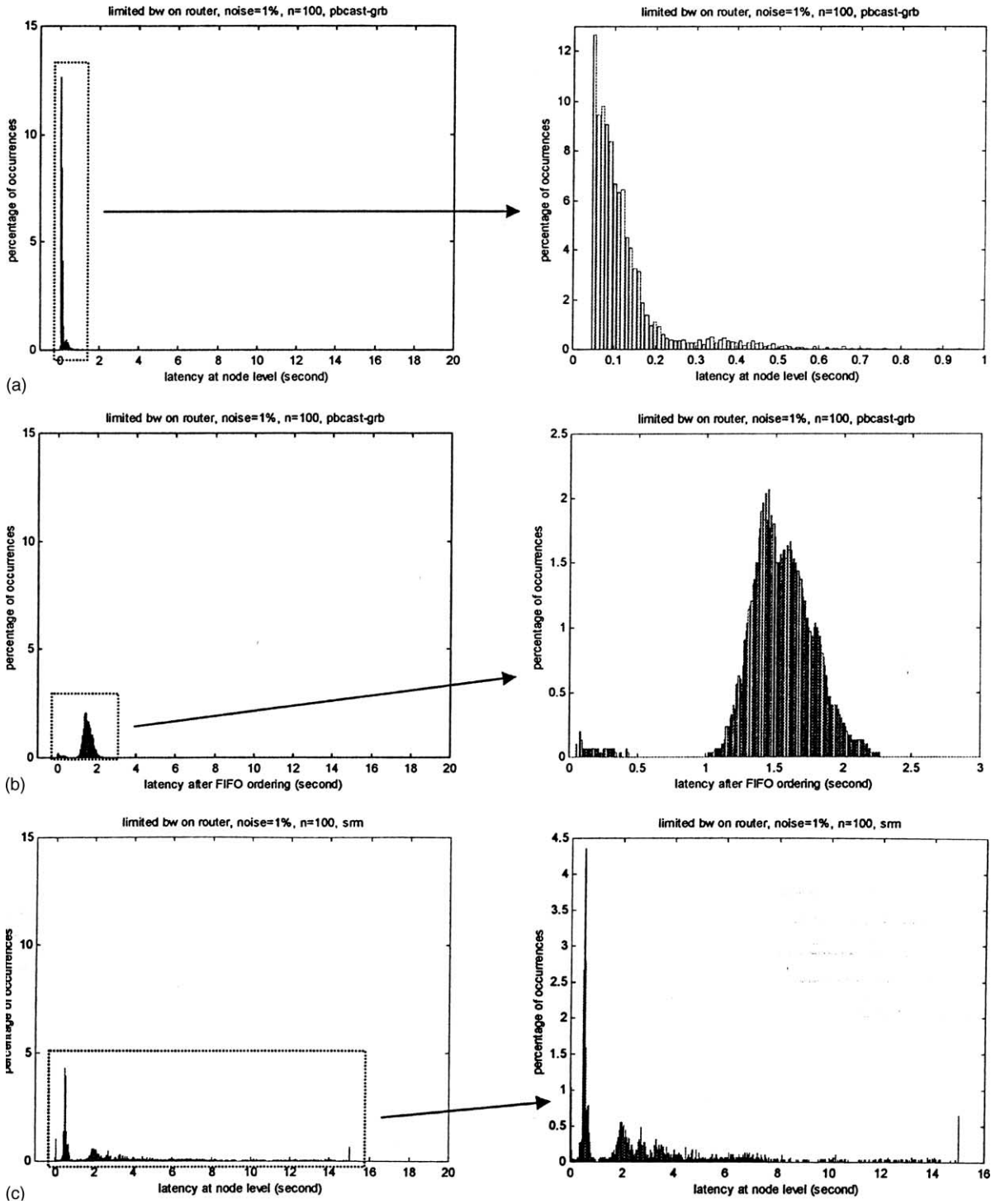

Fig. 13. Delivery latencies, 100-node tree topologies with limited bandwidth on root links. (a) Latencies for Bimodal Multicast at node level. (b) Latencies for Bimodal Multicast after FIFO ordering. (c) Latencies for SRM.

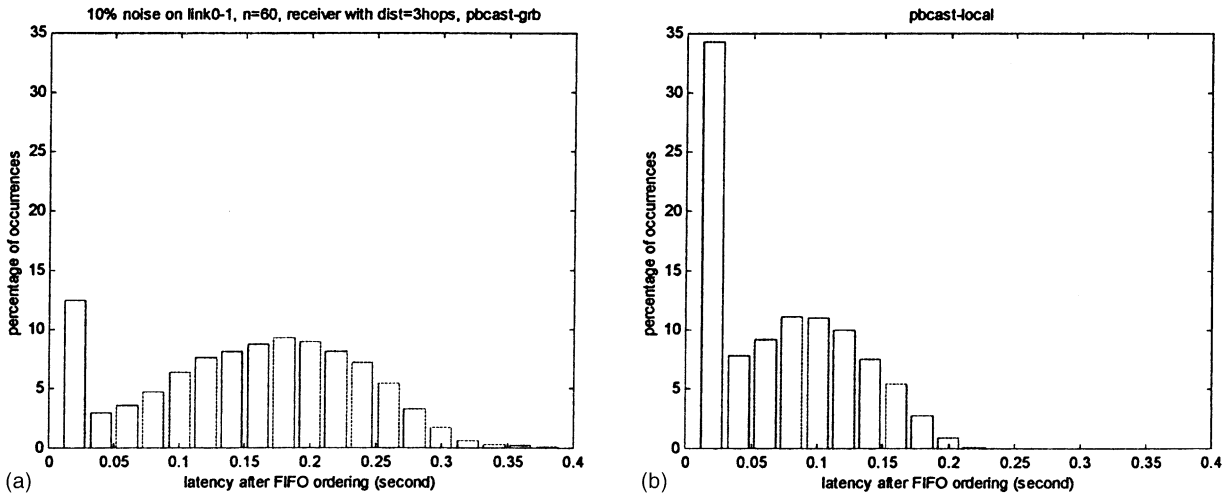

Fig. 14. Application level latency distributions of (a) Pbcast-grb, (b) Pbcast-local.

Behavior of a receiver process on the second cluster is analyzed to evaluate 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

$$\frac{1}{msgrate} + ld,$$

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/s and the total link delay from the sender to a particular receiver in the group is 40 ms, then $nd = 1/25 + 0.04 = 0.08$ s. We assume that messages received with latencies greater than 0.08 s are delayed because of the interference, and analyzed the percentage of data messages experiencing this delay.

Fig. 15 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 and 100 messages per second for Fig. 15(a) and (b), respectively. Fig. 16 shows the variation 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 and 20% for Fig. 16(a) and (b), respectively.

It is observed that, for most of the cases, 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 Bimodal Multicast.

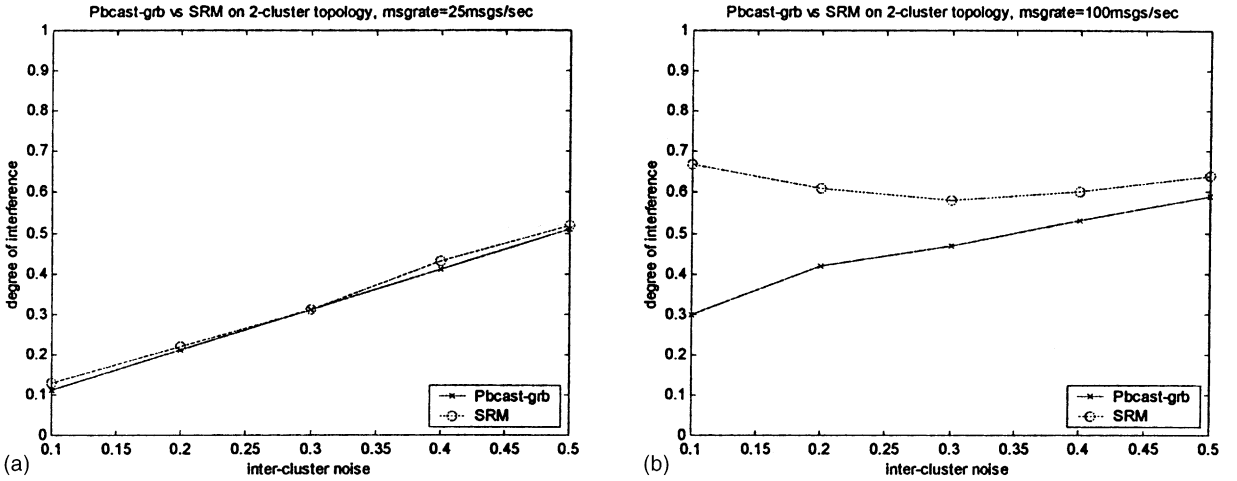

Fig. 15. Inter-cluster noise rate vs. degree of interference for Pbcast-grb and SRM. Multicast message rate: (a) 25 msgs/s, (b) 100 msgs/s.

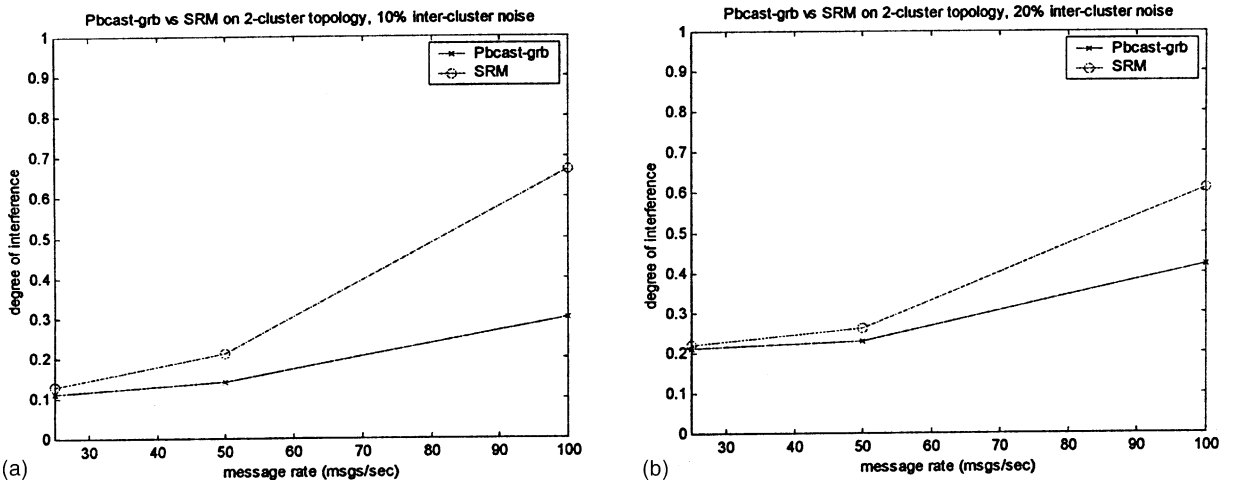

Fig. 16. Multicast message rate vs. degree of interference for Pbcast-grb and SRM. Inter-cluster noise rate: (a) 10%, (b) 20%.

6. Discussion

This study yields some general conclusions about the behavior of scalable reliable multicast protocols in distributed systems, some specific conclusions about the relative advantages and disadvantages of SRM and Bimodal Multicast, and also the limitations associated with each protocol.

Our simulation model enabled us to evaluate protocol performance on several network topologies, failure models, and group application scenarios. Furthermore, we compared Bimodal Multicast with SRM, offering best-effort reliability, and developed some optimizations to Bimodal Multicast. 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. In effect, SRM has higher bandwidth consumption compared to Bimodal Multicast protocols. We investigated the inter-arrival distributions of data messages for the protocols. Inter-arrival times of data messages are stable with an increase in the group size for Bimodal Multicast, 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. A detailed study of message latency behavior of the protocols was accomplished as well. Results show that, node-level message latencies of Bimodal Multicast is smaller compared to SRM's message latencies. We observed that as SRM is scaled to larger groups, steadiness of throughput could be expected to degrade. 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 were analyzed. It is demonstrated that high overhead rate can cause routers in a wide area network to become saturated easily.

It is observed 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 [18,19]. 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.

An issue about Bimodal Multicast 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. That would impact other applications sharing the network as well. As a remedy for this issue, hierarchical gossip mechanisms are explored in the Spinglass[1] project implementation of Bimodal Multicast and this is an area for further study. Initial results for hierarchical gossip strategy are discussed in [22].

Our studies show that Bimodal Multicast 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 IP multicast will successfully deliver most multicast data messages 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.

Unlike SRM, Bimodal Multicast 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

[1] http://www.cs.cornell.edu/Info/Projects/Spinglass/.

low protocol overhead. When IP multicast is used occasionally for retransmission as Pbcast-ipmc and Pbcast-grb do so, multicast data reaches most participants.

7. Conclusions

Inverted protocol stack is a new approach for overcoming throughput instability and scalability problems of traditional reliable multicast protocols. In this study, we demonstrated how the approach works well on several network settings by developing and using models for Bimodal Multicast exploiting an inverted protocol stack. We developed simulation models for Bimodal Multicast, and conducted extensive analysis studies for investigating protocol properties in practice and comparing it with best-effort reliable multicast mechanisms of SRM across various network characteristics and application scenarios. Results reveal that Bimodal Multicast and optimizations for improving its latency and reliability characteristics scale well, exhibit stable throughput and in contrast to the other scalable reliable multicast mechanisms, they provide predictable reliability even under highly perturbed conditions.

Acknowledgements

I am grateful to Professor Kenneth P. Birman from Cornell University, Department of Computer Science for his supervision, suggestions and comments during the development of this study.

References

[1] O. Babaoglu, R. Davoli, L. Giachini, M. Baker, Relacs: a communications infrastructure for constructing reliable applications in large-scale distributed systems, Technical Report, UBLCS-94-15, Laboratory for Computer Science, University of Bologna, Italy, 1995.
[2] S. Bajaj, L. Breslau, D. Estrin, et al., Improving simulation for network research, Technical Report 99-702, USC Computer Science Department, 1999.
[3] K.P. Birman, R. van Renesse, Reliable Distributed Computing with the Isis Toolkit, IEEE Computer Society Press, New York, 1994.
[4] K.P. Birman, Building Secure and Reliable Network Applications, Manning Publishing Company and Prentice-Hall, Greenwich, CT, 1997. http://www.browsebooks.com/Birman/index.html.
[5] K.P. Birman, M. Hayden, Ö. Özkasap, Z. Xiao, M. Budiu, Y. Minsky, Bimodal Multicast, ACM Trans. Comput. Syst. 17 (2) (1999) 41–88.
[6] S. Deering, D. Cheriton, Multicast routing in datagram Internetworks and extended LANs, ACM Trans. Comput. Syst. 8 (2) (1990) 85–110.
[7] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, D. Terry, Epidemic algorithms for replicated database maintenance, in: Proceedings of the Sixth ACM Symposium on Principles of Distributed Computing, Vancouver, BC, 1987, pp. 1–12.
[8] D. Dolev, D. Malki, The Transis approach to high availability cluster communication, Commun. ACM 39 (4) (1996) 64–70.
[9] S. Floyd, V. Jacobson, C. Liu, S. McCanne, L. Zhang, A reliable multicast framework for light-weight sessions and application level framing, IEEE/ACM Trans. Netw. 5 (6) (1997) 784–803.
[10] J. Gemmell, T. Montgomery, T. Speakman, N. Bhaskar, J. Crowcroft, The PGM Reliable Multicast Protocol, IEEE Network, January/February 2003.
[11] I. Gupta, K.P. Birman, R. van Renesse, Fighting fire with fire: using randomized gossip to combat stochastic scalability limits, J. Qual. Reliab. Eng. Int. 18 (3) (2002) 165–184 (special issue of quality and reliability of computer network systems).

[12] M. Hayden, The ensemble system, Ph.D. Dissertation, Department of Computer Science, Cornell University, 1998.
[13] S. Keshav, REAL: a network simulator, UCB CS Technical Report 88/472, 1988.
[14] V. Kumar, Mbone: Interactive Multimedia on the Internet, New Riders Publishing, Indianapolis, IN, USA, 1995.
[15] C. Labovitz, G.R. Malan, F. Jahanian, Internet routing instability, in: Proceedings of the SIGCOMM'97, 1997, pp. 115–126.
[16] R. Ladin, B. Lishov, L. Shrira, S. Ghemawat, Providing availability using lazy replication, ACM Trans. Comput. Syst. 10 (4) (1992) 360–391.
[17] K. Lidl, J. Osborne, J. Malcome, Drinking from the Firehose: Multicast USENET News, USENIX Winter 1994, pp. 33–45.
[18] C. Liu, Error recovery in scalable reliable multicast, Ph.D. Dissertation, University of Southern California, 1997.
[19] M. Lucas, Efficient data distribution in large-scale multicast networks, Ph.D. Dissertation, Department of Computer Science, University of Virginia, 1998.
[20] L.E. Moser, P.M. Melliar-Smith, D.A. Agarwal, R.K. Budhia, et al., Totem: a fault-tolerant multicast group communication system, Commun. ACM 39 (4) (1996) 54–63.
[21] Ö. Özkasap, Scalability, throughput stability and efficient buffering in reliable multicast protocols, Technical Report, TR2000-1827, Department of Computer Science, Cornell University, 2000.
[22] Ö. Özkasap, Z. Xiao, K.P. Birman, Scalability of two reliable multicast protocols, Technical Report, TR99-1748, Department of Computer Science, Cornell University, 1999.
[23] Ö. Özkasap, K.P. Birman, Throughput stability of reliable multicast protocols, in: Proceedings of the Advances in Information Systems (ADVIS2000), Lecture Notes in Computer Science, vol. 1909, Springer, Heidelberg, 2000, pp. 159–169.
[24] S. Paul, K. Sabnani, J.C. Lin, S. Bhattacharyya, Reliable multicast transport protocol (RMTP), IEEE J. Select. Areas Commun. 15 (3) (1997) (special issue on network support for multipoint communication).
[25] V. Paxson, End-to-end Internet packet dynamics, in: Proceedings of the SIGCOMM'97, 1997, pp. 139–154.
[26] R. Piantoni, C. Stancescu, Implementing the Swiss Exchange Trading System, FTCS 27, Seattle, WA, 1997, pp. 309–313.
[27] A.S. Tanenbaum, M. van Steen, Distributed Systems: Principles and Paradigms, Prentice-Hall, Upper Saddle River, NJ, 2002.
[28] R. Van Renesse, K.P. Birman, S. Maffeis, Horus: a flexible group communication system, Commun. ACM 39 (4) (1996) 76–83.
[29] W. Vogels, R. van Renesse, K.P. Birman, The power of epidemics: robust communication for large-scale distributed systems, Comput. Commun. Rev. 33 (1) (2003) 131–135.

Öznur Özkasap received her M.Sc. and Ph.D. degrees in computer engineering, both from Ege University, Izmir, Turkey, in 1994 and 2000, respectively. Since September 2000, she is an Assistant Professor in the Department of Computer Engineering of Koc University in Istanbul, Turkey. During her Ph.D., she was awarded the TÜBITAK-NATO A2 Ph.D. Research Scholarship Abroad and spent 1997–1999 at Cornell University, Department of Computer Science, Ithaca, NY, where she completed her Ph.D. Dissertation. From 1992 to 2000, she has been a Research and Teaching Assistant at Ege University, Department of Computer Engineering, and from 1997 to 1999 she has been a Graduate Research Assistant at Cornell University, Department of Computer Science. Her research interests include distributed computing systems, scalable reliable multicast protocols, peer-to-peer communication, protocol performance evaluation, parallel computing, fault-tolerant distributed systems, distributed real-time systems, and computer networks.