Evaluation of Network Latency Impact on Atomic Broadcast Performance

David Bakken*, Jin Ding, Krishna Sivalingam, and Limin Gu

School of EECS, Washington State University, Pullman, Washington USA

{bakken,lgu,jding1,krishna}@eecs.wsu.edu

April 12, 2002

Abstract

Reliable Atomic Broadcasts (ABCASTs) do not scale well over wide area networks (WANs) due in part to operational conditions such as higher packet drop rates and network latencies. However, while much work has been done optimizing such protocols for local area network (LAN) environments, there has been no work to date on systematically quantifying the effects of network conditions on ABCAST performance, in order to better understand how and why they do not scale to WANs and to suggest new protocol directions. This paper presents results from a systematic study of how multicast performance is affected by latency, latency variance, number of senders, traffic types.

Keywords: group communication, reliable multicast, atomic broadcast, virtual synchrony.

1 Introduction

Virtually synchronous group communication systems provide powerful building blocks on which to construct replicated services [6]. The main service required by such systems is a reliable and totally-ordered atomic broadcast (ABCAST). However, the performance of ABCASTs can be affected by many environmental conditions, hardware configurations, and other operational factors the protocols cannot control, and in some cases cannot even ascertain or predict at runtime. Such factors include network throughput and latency; network packet sizes and drop rates; router buffer sizes; the size and sending rate of multicast message; CPU loads; and the group size. Adverse values for these factors can greatly reduce multicast performance metrics such as the throughput and latency of multicast delivery, message stability times, failover time, and the amount of buffering required at group members. (Unfortunately, the literature seems devoid of a systematic study of the effect of such factors on these performance metrics.)

*Contact author. Phone: +1 509 335 2399. FAX: +1 509 335 3818. Address: PO Box 642752; Pullman, WA 99164-2752; USA.

For these reasons, virtually synchronous group communication systems do not scale well over wide area networks (WANs). Their higher values of network latencies clearly have an adverse impact. However, we believe that it is the higher variance in these network latencies that is at least as harmful to performance, for two reasons. First, such variance makes it difficult, and arguably impossible, to set a workable timeout value for the failure detectors. Second, this variance increases the dispersal of the arrival of messages in a given round. In many protocols, group members must wait for all messages to arrive before proceeding, for example to ensure atomicity, so such network latency variance directly impacts the delivery latency and throughput of the multicasts.

In this paper we report on one such investigation. The contribution of this paper is a systematic study of how multicast performance is affected by latency, latency variance, number of senders, traffic types, and packet drop rate. These are done for both a sequencer- based and a token-based ABCAST, as well as for pbcast.

The remainder of this paper is organized as follows. Section 2 provides background information in multicast protocols. Section 3 presents our study on the effect of operational parameters on multicast performance. Section 4 discusses related work, and Section 5 summarizes the paper and discusses future work we plan.

2 Background

Total order multicast algorithms can fall into three basic classes, depending whether the order is built by sender, sequencer, or destination processes respectively [28]. Since there are differences in the algorithms among those three basic classes, five subclasses can be defined: communication history, privilege-based, moving sequencer, fixed sequencer, and destinations agreement. Privilege-based and moving sequencer are also referred as token-based algorithms.

In communication history, the order is built by sender processes. A partial order $<$ is defined on messages based on the history of communications. Communication history algoirthms deliver messages in a total order that is compatible with the partial order $<$. Privilege-based algorithms rely on the idea that senders can multicast messages only when they are granted the previlege to do so. There is a cyclic order among group senders and a token circulates in this order. When a group member wants to multicast a message, it must wait until it holds the token. Moving sequencer algorithms are based on the idea that a group of processes successively act as sequencer. It is similar to the Privilege- based algorithms, but it is different in that the sequencer is related to the destinations rather than the senders. In a fixed sequencer algorithm, a designated sequencer responsible for ordering messages. There is only one such sequencer in the group, and the responsibility is not normally transferd to other members. In desination agreement algorithms, the order results from an agreement between destination processes. To multicast a message, a sender send it to all destinations, and all destinations assign the received message a local timestamp and multicast this timestamp

to all destinations. Once a destination process has received a local timestamp from all other destinations, the message can be delivered.

Even though all these multicast algorithms are well known and have been studied for years in the group communication research community, most of those studies were focused on the optimization of the protocols. In [11], Cristian and Mishra conducted an interesting comparison study about token-based and sequencer-based atomic broadcast protocols. Five performance metrics: average delivery time, average stability time, average number of physical messages sent per update broadcast, maximum buffer size, and distribution of processing load among group members are measured as a function of group size and update inter-arrival time, both in the absence of failures and in the presence of a single communication failure. Over LANs, multicast group communication protocols now achieve throughputs for reliable totally ordered multicasts that exceed the throughput that TCP achieves for point-to-point communication. [1] We note that this improvement over TCP is possible, despite ABCASTs providing stronger guarantees, because they provide the guarantees (and do retransmissions and keep state) at the message level, not the byte level. However, since those broadcast or multicast protocols are originally designed for applications in LANs, network latency and its variance are never systematically studied as the operational factors on performance.

There are also numerous new research attempts on group communication systems which are designed for WANs [2, 20]. The protocols struggle to provide performance and most of them use relatively weaker semantics over the Wide-Area Networks. While such weakened semantics help increase performance, they are much harder to program with, negating a key advantage of the virtual synchrony model supported by ABCASTs.

Clearly, one of the fundamental problems is that the protocols face more unpredictable and larger network latency. In order to get a better understanding of the impact of network latency when the virtually synchronous group communication system scales up to WAN applications, a systematic study about the performance of multicast protocols under varying network, application and group configuration is desirable. Such information can also be very useful for tuning ABCAST protocols at runtime, and are also crucial for protocol switching [21].

3 Impact of Network and Application Operational Parameters on Atomic Broadcast Performance

In this investigation we implemented three multicast algorithms: privilege-based, fixed sequencer and destinations agreement, which are selected from each of three basic classes in the Network Simulator (NS-2) package. The privilege-based algorithm, which is also refered as token-based protocol, is similar to the single ring totem protocol [3]. The fixed sequencer algorithm, which is also referred as sequencer-based,

[1] **NOTE**: Can we back this claim up (I think so)? If so, Jin, please provide a citation and update this. Dave 4/11.

is similar to ensemble sequencer-base protocol [18]. Destination agreement is similar to pbcast protocol [5]. So in the following, we refer them as sequencer-based, token-based and pbcast protocols. We note that the pbcast protocol studied provides weaker guarantees than the other two, and no virtual synchrony. It is studied here for contrast with the other two, as a (likely unachievalbe) upper bound on virtually synchronous performance, because pbcast has been designed for much better performance in WANs, especially higher and more stable throughput.

The following assumptions were made in the simulation model:

- The network is assumed as a virtually fully connected network, where every node has a virtual link (i.e. multi-hop path in real topology) to every other node.
- Each virtual link has an associated latency. We assume the network latency varies according to certain distribution during each simulation. We did not utilize the IP multicast facility provided in NS-2, since it does not allow the latencies of delivery to each destination to vary; i.e. all recipients suffer the same latency which is not desired for our purposes.
- Point-to-point messaging is assumed because our goal is to measure the overall performance of those multicast protocol in a generic communication environment.
- The past 15 years of Internet traffic measurement have produced key observations [30]. Although characterizing aggregate network traffic is difficult, network researchers have identified a significant degree of long-range dependence (LRD) in network traffic, which they refer to as ”self-simliar,” ”fractal,” or ”multifractal” behavior. Self-simliar model can be obtained by the superposition of many ON/OFF sources in which the ON and OFF periods have a Pareto type distribution with infinite variance [31]. Beside choose $CBR_Traffic$ which generates traffic according to a deterministic rate, we choose $POO_Traffic$ which generates traffic according to a Pareto On/Off distribution.
- In order to understand the impact of network latency distribution, we chose three types of distributions: normal, log-normal, exponential. Figure 1 illustrate the probability density function (PDF) of these three distribution types on different mean values.
- No packet are assumed to be dropped by network.

The system parameters are varying network latency distribution, mean network latency, latency variance, the number of concurrent senders. Table 1 shows the parameters used in our simulations. [2]

The performance metrics studied are avarage message stability times and the message stability time distributions which are presented in the form of a probability density function (PDF). Message stability

[2]**NOTE**: Jin: please add defaults in parentheses to the table, and make them bold, as I have changed the header to be. Also, discuss with Krishna if the 3 latency parameters should have lines between their rows or not — I don't care either way. Finally, I read somewhere that a caption for a table is supposed to above the table, unlike a figure, so if you can find a few examples either way in this or other IEEE journals, then discuss this with Krishna. Dave 4/12.

Parameters	Values (**defaults**)
The number of members in the group	10
The number of senders in the group	1, 2, 3, 4, 5, 6, 7, 8, 9,10
Network latency distribution Mean network latency Latency variance	Normal, Log-Normal, Exponential 0.25sec, 0.5sec, 1sec, 2sec 0.25, 0.5, 0.75, 1
Traffic type (1)	CBR with 1Mb/sec
Traffic type (2)	Pareto On/Off with burst time 1000ms, idle time 1000ms, 1Mb/sec
Packet size	1000KB

Table 1: Operational parameter variables for multicast protocols simulations.

time is defined as the duration between the moment that a sender receives a multicast message from the application to the moment that sender learns all the receivers receive the message. At this point, the final ordering of the message is known, it is eligible to be delivered to the application, and the message can be removed from the sender's buffer. While the average message stability time is useful performance metric, the distribution of message stability times is also an important consideration for real-time distributed system. Throughout the simulations in this paper, the message stability times are collected from over 1,250,000 messages originated from each sender. Results of other typical metrics such as average message delivery time and throughput are not presented since we found that average message stability times are reasonably proportional to average message stability times in our simulations, and the throughput is not the concern of our configurations.

3.1 Impact of Mean Network Latency

The purpose of this experiment is to investigate how the mulitcast protocols scale with the network latency. Because the results of the CBR traffic are almost the same as that of Pareto On/Off, we only present the results of the CBR traffic. Figure 2, figure 3, figure 4 presents the average message stability time against mean network latency in sequencer-based protocol, token-based protocol, and pbcast protocol with 1 sender and 10 senders in the group. The mean network latency varies from 0.25 seconds to 2 seconds. First, we can see that different network latency distributions got different average message stability time, and the difference increases as the mean network latency increases. For all these three protocols, the average message stability time increased linearly as the mean network latency increases. But token-based protocol increases faster and has higher average stability time than the other two protocols. This is expected, because the performance of token-based protocol depends on a single improtant message the token's latency.

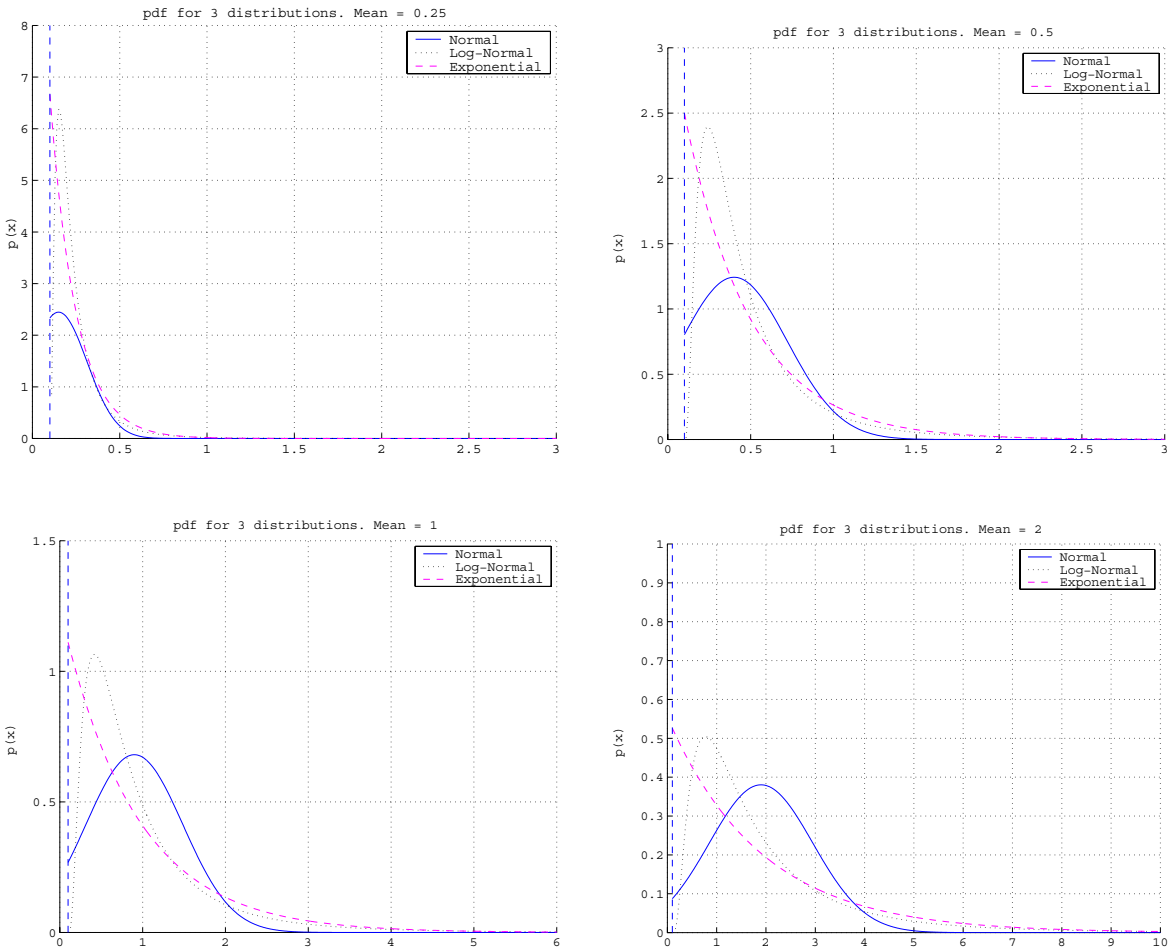


Figure 1: Probability Density Function of Selected Distributions.

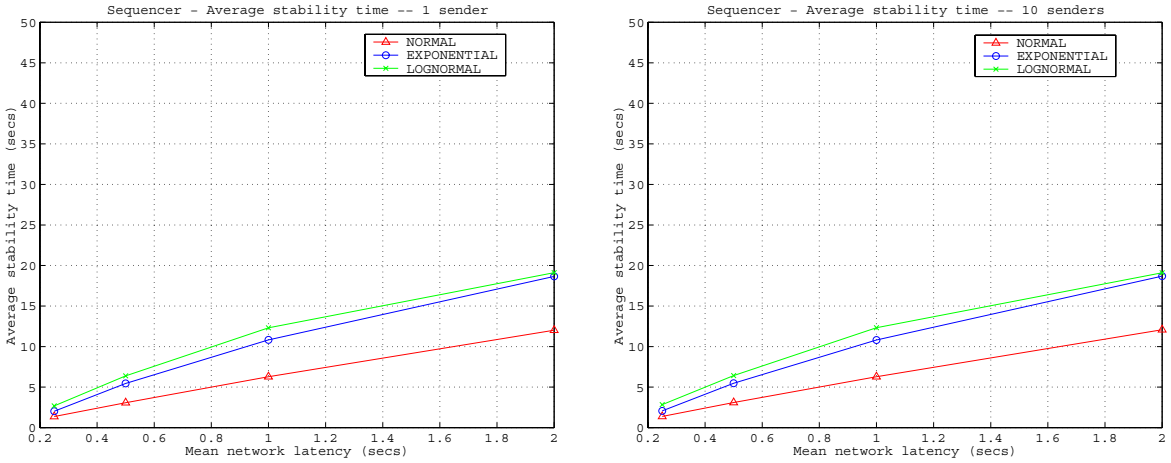


Figure 2: Average message stability time for sequencer-based protocol with $meannetworklatency \in \{0.25s, 0.5s, 1s, 2s\}$, $sender \in \{1, 10\}$.

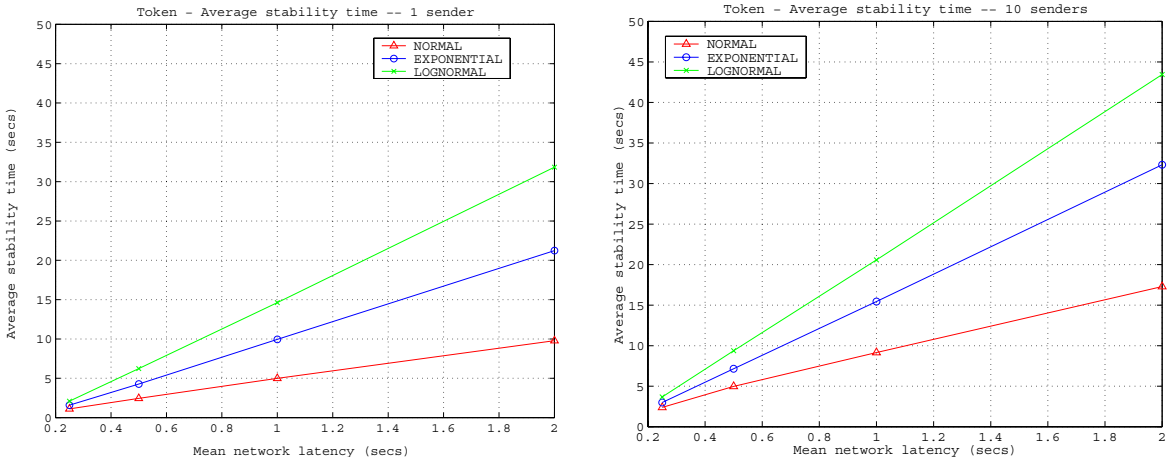


Figure 3: Average message stability time for token-based protocol with $meannetworklatency \in \{0.25s, 0.5s, 1s, 2s\}$, $sender \in \{1, 10\}$.

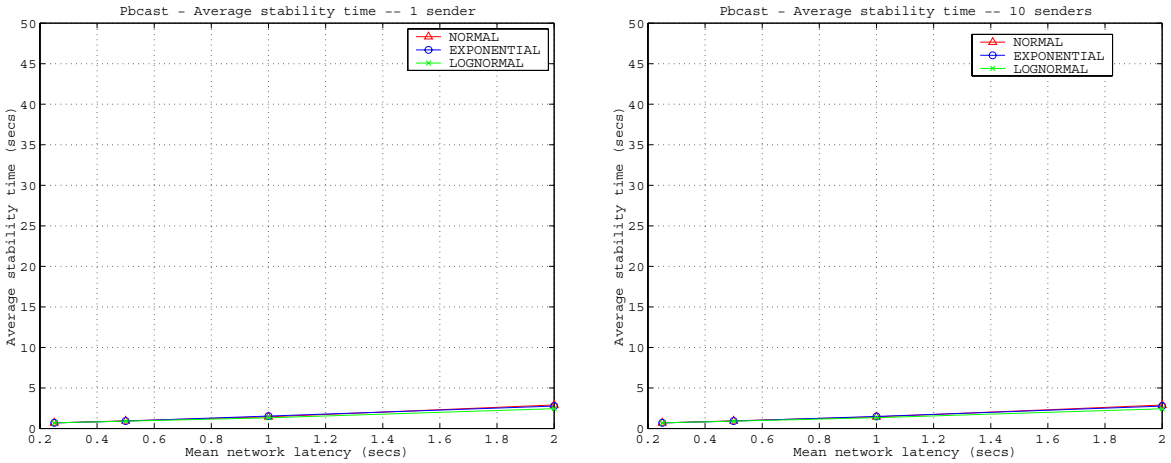


Figure 4: Average message stability time for pbcast protocol with $meannetworklatency \in \{0.25s, 0.5s, 1s, 2s\}$, $sender \in \{1, 10\}$.

3.2 Impact of Latency Variance

In this section, we study how the network latency variance impact the performance of the multicast protocols. Because the results of CBR traffic is almost the same as that of Pareto On/Off, we only presents the results of CBR traffic. Figure 5 presents average stability time of sequencer-based protocol, token-based protocol and pbcast protocol with mean network latency is 0.5s and 10 senders in the group. network latency variance varies in 0.25, 0.5, 0.75, 1. Normal and lognormal distributions are selected. For pbcast protocol, the distribution and variance of network latency has little impact on the average stability time. But they did have impact in sequencer-based protocol and token-based protocol. The average message stability time increases as the variance increases in these two protocols, and increases much faster in token-based protocol than in sequencer-based protocol.

3.3 Impact of Number of Senders

In this section we look at the number of senders affect the performance of multicast protocols. Figure 6 presents the message stability time distributions for sequencer-based protocol, figure 7 for token-based protocol, and figure 8 for pbcast protocol. (a) (b) is for CBR traffic, (c)(d) is for Pareto On/Off traffic. The mean network latency is 0.5 second, and there are 1 sender and 10 senders in the group. Normal, lognormal, and exponential distributions are selected. From the figures we can see that the message stability time distributions are slight different in CBR traffic and Pareto On/Off traffic. In each protocol, different message stability time distributions are got against different network latency distributions. For sequencer-based protocol, and token-based protocol, the message stability time distribution in lognormal span longer range than the other two distributions. It is also seen that with the increase of the number of senders, different net-

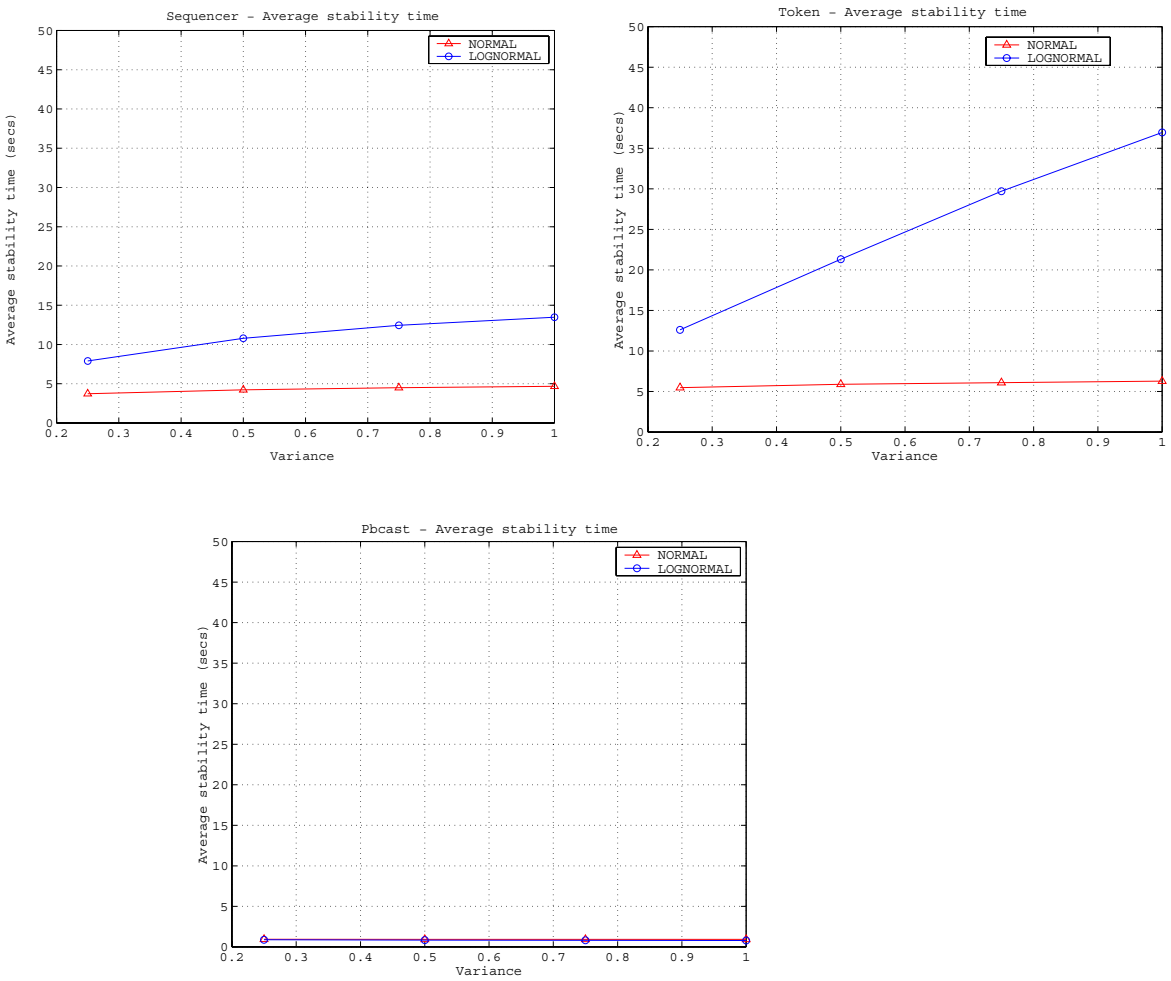


Figure 5: Average stability time for sequencer-based protocol, token-based protocol and pbcast protocol with network latency mean value = 0.5s, $distribution \in \{normal, lognormal\}$, sender = 10, varying $variance \in \{0.25, 0.5, 0.75, 1\}$.

work latency distributions had different message stability time distributions in sequencer-based protocol and token-based protocol, and got same message stability time distributions in pbcast protocol. It is expected because message stability time in pbcast protocol depends on gossip interval and gossip threshold. Even with the same mean network latency, the average message stability time in each distributions is different. For pbcast protocol, the difference of the message stability time distributions are not obvious[3] due to the message stability time is mostly determined by gossip interval and gossip threshold.

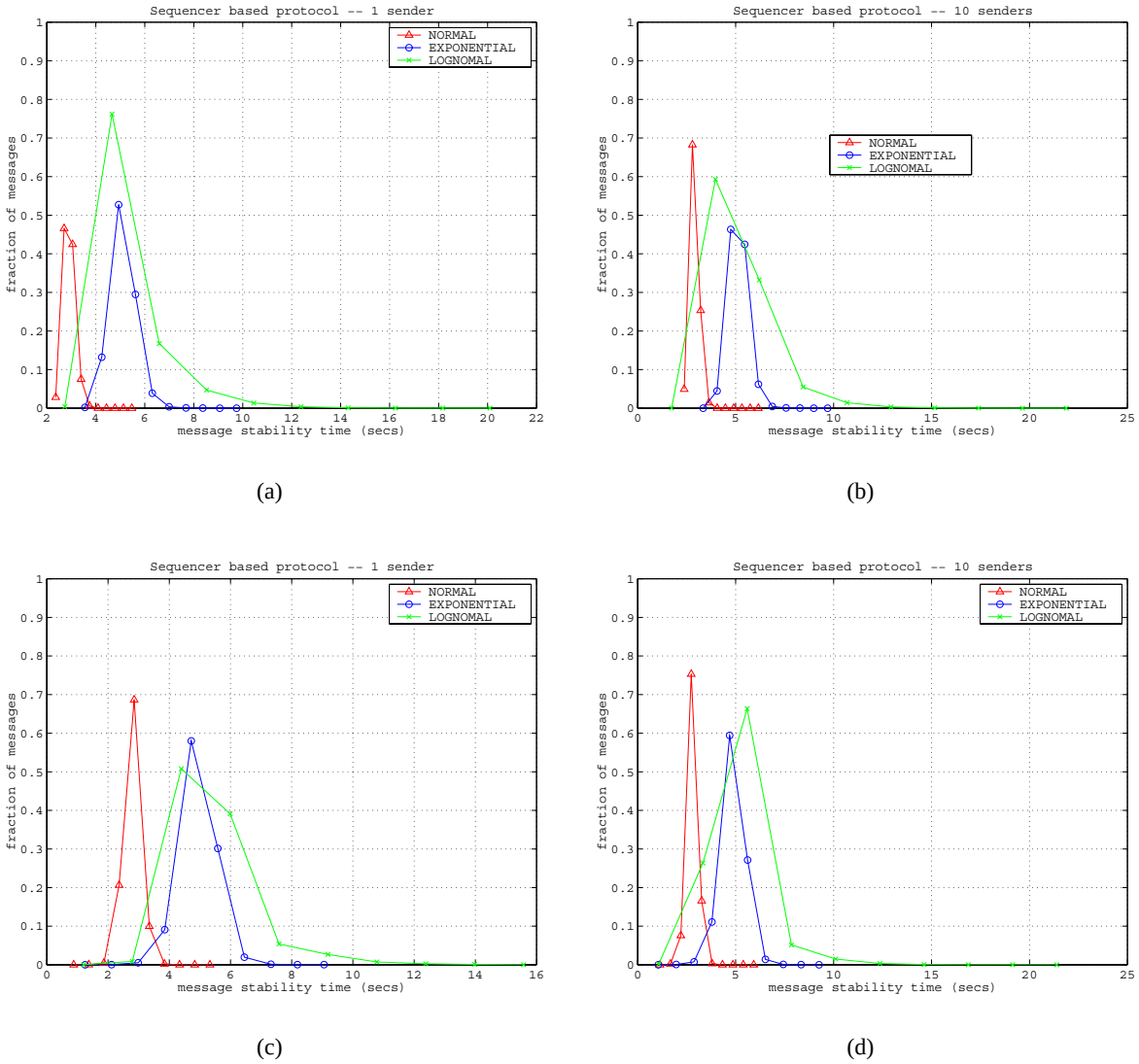


Figure 6: Message Stability time distributions for sequencer-based protocol with mean network latency = 0.5 s, $sender \in \{1, 10\}$. (a)(b) for CBR traffic, (c)(d) for Pareto On/Off traffic

[3]**NOTE**: Krishna: "obvious" or "clear cut" etc. probably needs to be modified — the wording sounds like it is not obvious to us cause we are not smart or something. Maybe something more diplomatic; for this example regarding pbcast is is because we woudl have to run an entire series of experiments with different gossip intervals and gossip thresholds. Dave 4/12.

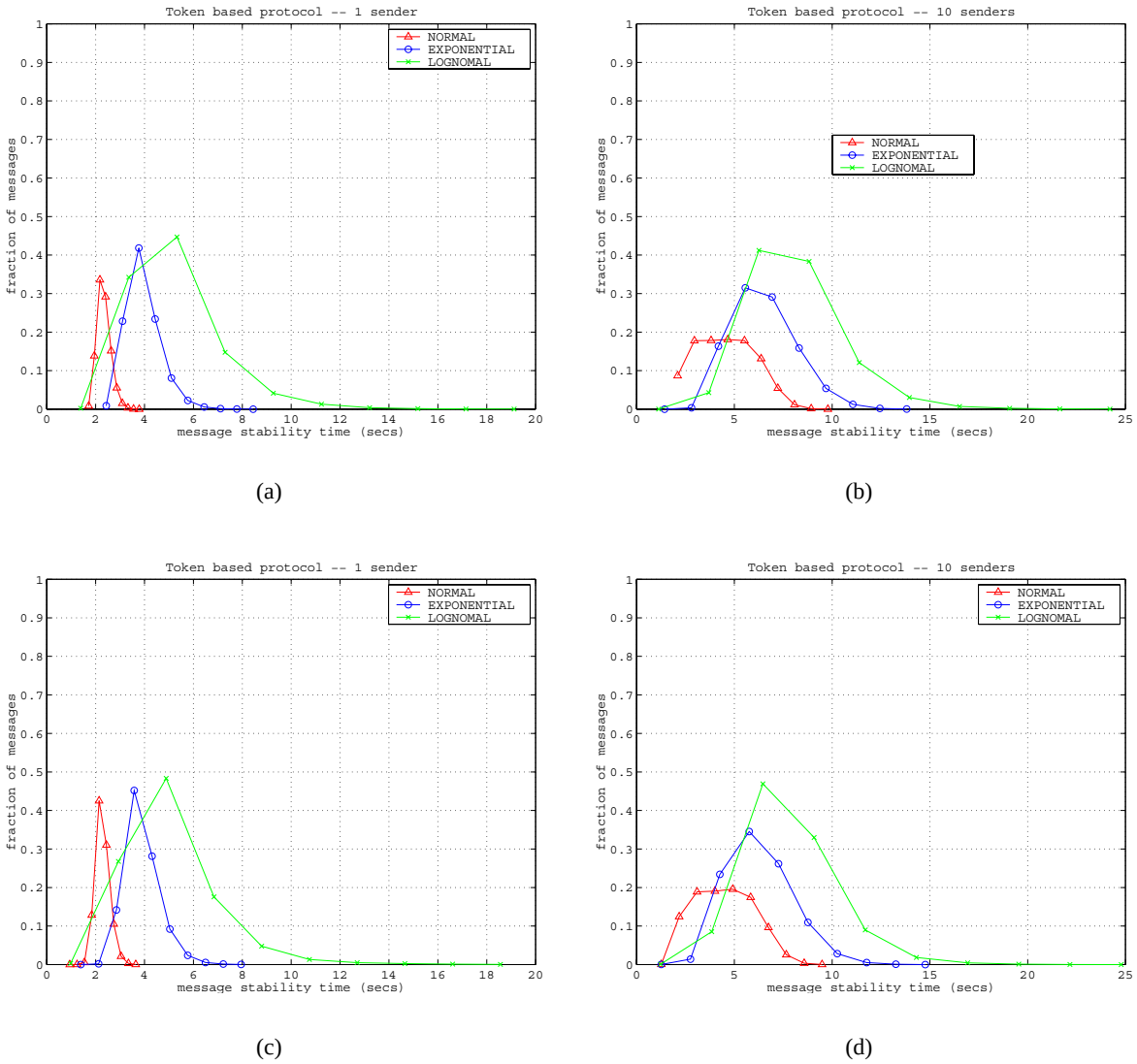
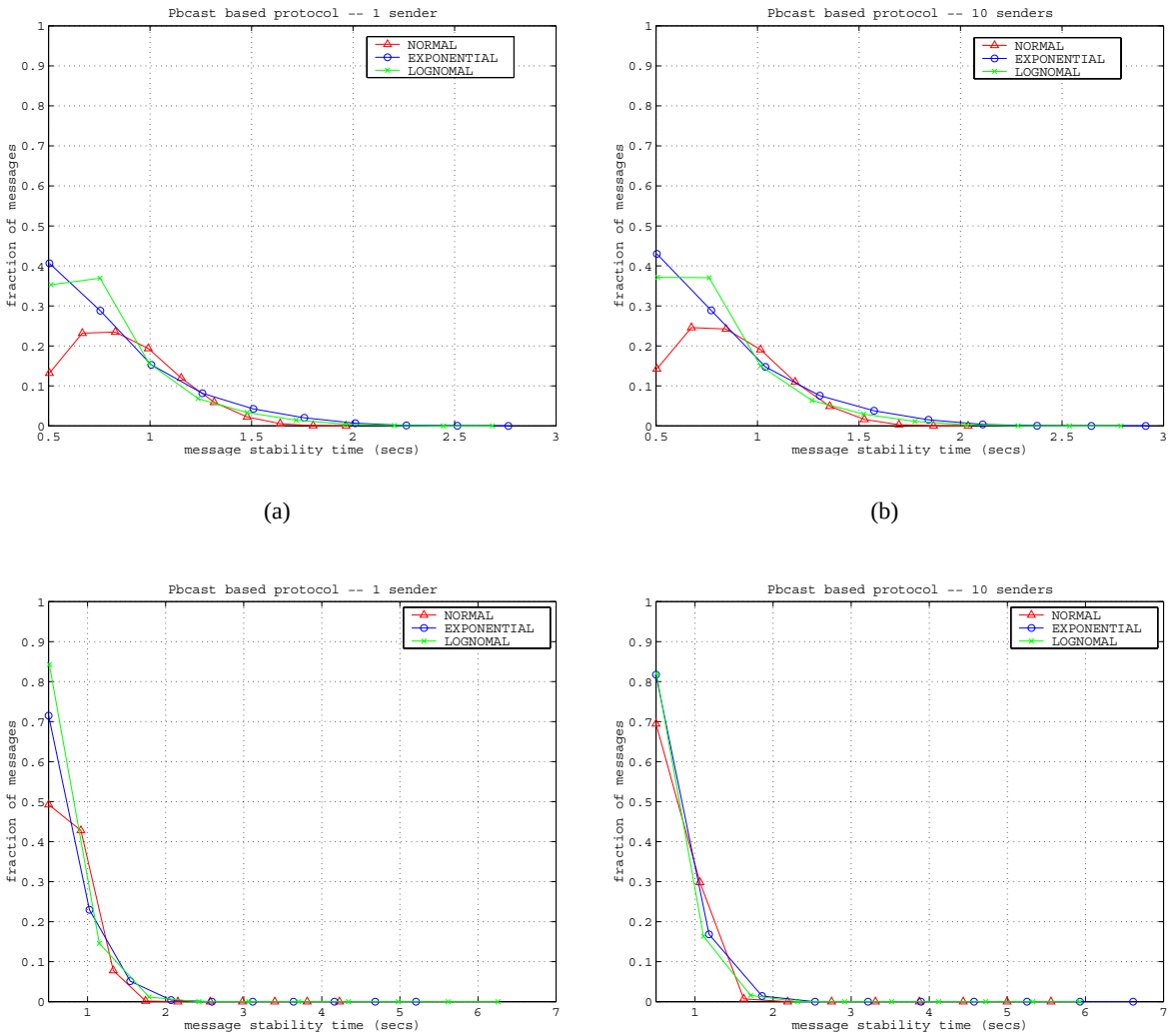


Figure 7: Message Stability time distributions for token-based protocol with mean network latency = 0.5 s, $sender \in \{1, 10\}$. (a)(b) for CBR traffic, (c)(d) for Pareto On/Off traffic

Figure 9 presents the average stability time for sequencer-based protocol, token-based protocol and pbcast protocol with network latency mean value is 0.5s, traffic generation rate is CBR. The reason that we only present the results of CBR traffic is that it is almost the same as that of Pareto On/Off. The number of senders in the group varies from 1 to 10. In the same mean network latency, the average message stability time in each distributions is different. For pbcast protocol, the difference of the message stability time distributions are not obvious due to the message stability time is mostly determined by gossip interval and gossip threshold. In addition, with the increase of the number of senders, average message stability times have no changes in sequencer-based protocol, and increase linearly in token-based protocol. This is because the token-based protocol disallows concurrent sending by senders. In token-based protocol, only sender,



(a) (b)

(c) (d)

Figure 8: Message Stability time distributions for pbcast protocol with mean network latency = 0.5 s, $sender \in \{1, 10\}$. (a)(b) for CBR traffic, (c)(d) for Pareto On/Off traffic

which has the token, can send the message. Thus the message stability time in token-base protocol depends on the token's latency. With the increase of the number of senders, the token's latency will also increase.

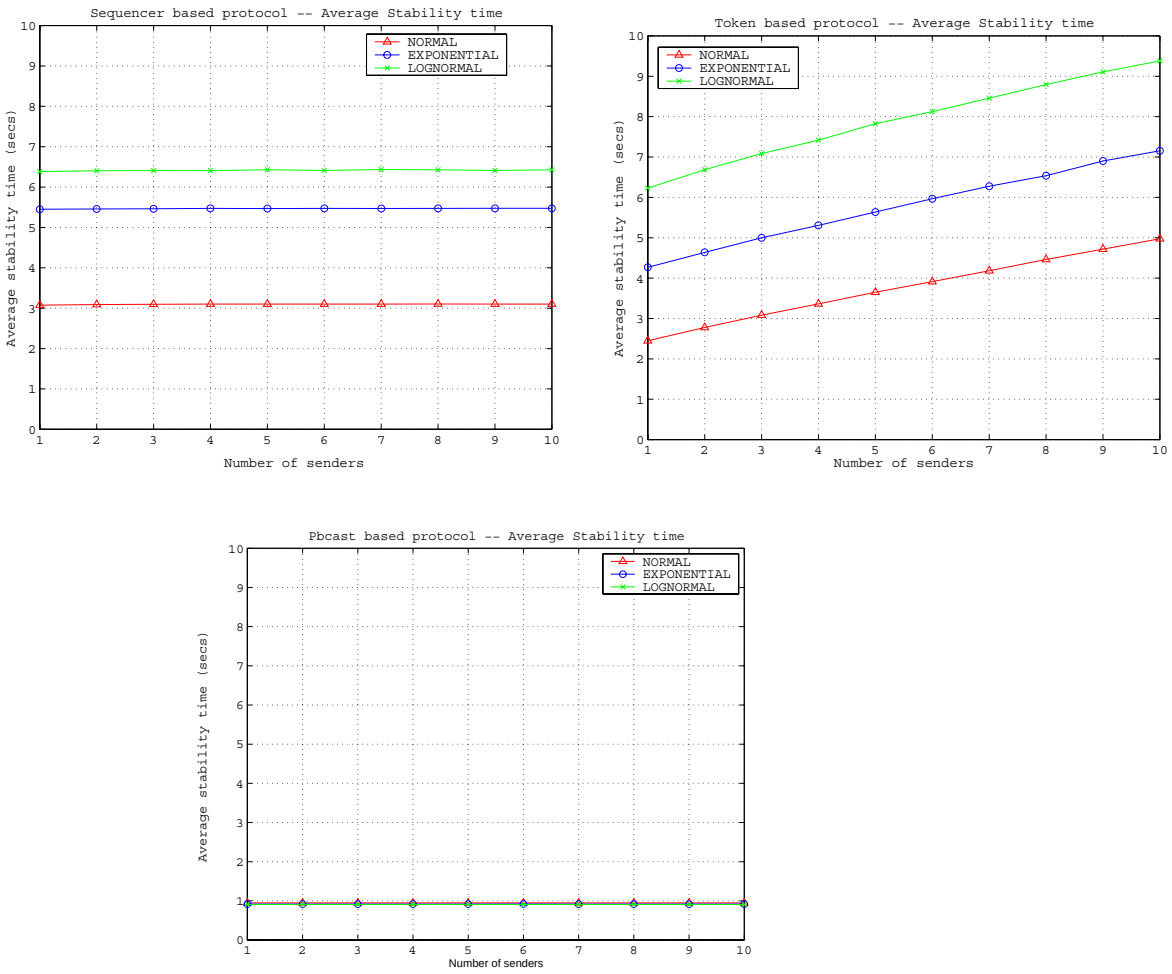


Figure 9: Average stability time for sequencer-based protocol, token-based protocol and pbcast protocol with network latency mean value = 0.5s, $distribution \in \{normal, lognormal, exponential\}$.

3.4 Summary of Results

From above, we can see that the distributions of network latency did have impact on the distributions of message stability time and average message stability time in group applications: sequencer-based protocol, token-based protocol, and pbcast protocol. But the impact is different upon different protocols. For pbcast protocol, only small change occurred, while else big changes happened in token-based protocol. This can explain the reason that pbcast is designed to be scalable across a WAN, while token-based protocol is not. This scalability is of course at the cost of the stronger properties that the token-based protocol supports. So given that the current statistical characteristic of network latency over wide-area networks has not been

agreed upon by the Internet research community, the further study left to us is to find if the bandwidth reservation techniques can improve the performance of group applications over WAN.

4 Related Work

We are aware of no work evaluating the costs of network latencies or the other parameters we have studied on multicast latencies. [4] [7] presents new totally ordered multicast algorithms that guarantee an upper bound on the maximum latency on message delivery to the application in certain network models. The paper considers two sender based reservation models: constant bit rate (CBR) where application reserves a fixed bandwidth, and variable bit rate (VBR) where application reserves average transmission rate and a maximum burst time over a long period of time. For these models, the authors prove their algorithms preserve the bandwidth and latency reserved by the application within a certain additive constant. However, [7] only proves that totally ordered multicast can exist with an upper bound on message delivery latency with the help of reservation. On the other hand, this paper investigates how two basic categories of totally ordered multicast protocols, sequencer-based and token-based, subject to various network and application operational parameters.

[8] is very complementary work to ours. It presents a detailed theoretical study of their new ABCAST algorithm under different circumstances. Their algorithm provides multicast latency that is within a constant of the latency of the underlying network, independent of the number of participants. They assume QoS and then build the algorithm to work under this assumption. In contrast, we focus on algorithms that work without QoS, and try to quantify where they have performance problems in WAN-like environments, to help quantify exactly where they break down. We also do not analyze membership changes, which they do.

[5] introduces a bimodal delivery guarantee as falling between expensive and poorly scalable ABCAST protocols, and inexpensive and scalable best-effort delivery. The article argues that stability of multicast delivery is a basic reliability property, i.e., a multicast protocol is reliable in a sense that can be rigorously quantified and includes throughput stability guarantees. The theoretical analysis and experimental results presented in the paper confirm that the bimodal multicast is reliable and scalable and provides remarkably stable delivery throughput.

Budhia discusses several different approaches for achieving high performance in group communication systems, using clever flow control algorithm, careful system design and engineering, and proper system tuning [9] based on the Totem system [1]. Also performance graphs in terms of packet throughput and message delivery latency, for different network topologies and under different operating conditions are presented. However the impacts of network and application operational parameters as presented in this paper are not considered in that dissertation.

[4] **NOTE**: Krishna: please rephrase this first sentence to please performance analysis folks. Thanks, Dave 4/12.

5 Conclusions and Future Work

In this paper we have presented a simulation study that quantified the effect of the following operational characteristics on multicast performance: network latency statistical distribution, network latency variance, number of senders, application traffic type, and network packet drop rate. These results are summarized in Section 3.6.

This research has just scratched the surface in an area crucial to help virtually synchronous multicasts scale better over WANs, and we hope to build on it in many ways. Expanding it to include more protocols is important: a few other virtually synchronous protocols, at least [8], and perhaps also pinwheel [14] or newsmonger [22]; a few virtually synchronous protocols that provide dynamic uniformity; a protocol or two such as [25, 12] that tolerate Byzantine failures; well as the (non virtually synchronous) pbcast protocols [5]. We also hope to expand simulations to include group membership changes and the multicast performance metrics measured to include failover time. The large variance in WANs of course makes setting workable timeouts very difficult, and quantifying the benefit of lower and less variant latencies towards reducing failover time (especially failure detection) seem promising. We hope to investigate how well a few experimental bandwidth reservation protocols provide latency. We also hope to combine bandwidth reservation and multicast protocols, building on [17].

Acknowledgements

To be added later.

References

[1] Deborah A. Agarwal. ”Totem: A Reliable Ordered Delivery Protocol for Interconnected Local-Area Networks”, Ph.D. Dissertation, University of California, Santa Barbara, December 1994.

[2] Y. Amir, C. Danilov, and J. Stanton. ”A Low Latency, Loss Tolerant Architecture and Protocol for Wide Area Group Communication”. The International Conference on Dependable Systems and Networks, Pages 327-336, New York, June 2000.

[3] Y. Amir, L. Moser, P. Melliar-Smith, D. Agarwal, and P. Ciarfella. ”The totem single- ring ordering and membership protocol”, ACM Transactions on Computing Systems, Vol. 13, No. 4, Pages 311-342, 1995.

[4] S. Blake, D. Black, M. Carlson, E. Daies, Z. Wang, and W. Weiss. ”An Architecture for Differentiated Services”, RFC 2475. December 1998.

[5] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. ”Bimodal Multicast” ACM Transactions on Computer Systems, Vol. 17, No. 2, May 1999, Pages 41-88

[6] A Review of Experiences with Reliable Multicast. Kenneth P. Birman. Software, Practice and Experience 29(9), 741-774 (Sept. 1999).

[7] Z. Bar-Joseph, I. Keidar, T. Anker, and N. Lynch, ”QoS Preserving Totally Ordered Multicast”, to appear in Proceedings of the 5th International Conference on Principles of DIstributed Systems (OPODIS), Paris, France, December 2000.

[8] Bar-Joseph, I. Keidar, and N. Lynch, ”A Fault-Tolerant Dynamic Atomic Broadcast Algorithm with QoS Guarantees”, submitted to DSN-2001.

[9] Ravi K. Budhia. ”Performance Engineering of Group Communication Protocols”. Ph.D. Dissertation, University of California, Santa Barbara, August 1997.

[10] G. Ballintijn, and M. van Steen. ”Characterizing Internet Performance to Support Wide-area Application Development”, ACM Special Interest Group on Operating Systems, Vol. 34, No. 4, October 2000.

[11] F. Cristian, R. de Beijer, and S. Mishra. ”A Performance Comparison of Asynchronous Atomic Broadcast Protocols”, Distributed Systems Engineering, Vol. 1, No.4, Pages 177-201, June 1994.

[12] M. Castro and B. Liskov. ”Practical Byzantine Fault Tolerance”, Proceedings of the Third Symposium on Operating System Design and Implementation, ACM, February 1999.

[13] J. M. Chang and N. F. Maxemchuk. ”Reliable Broadcast Protocols”, ACM Transaction on Computer Systems, Vol. 2, No. 3, August 1984, Pages 251-273.

[14] F. Cristian and S. Mishra. ”The Pinwheel Asynchronous Atomic Broadcast Protocols”, Proc. 2rd Int. Symp. On Autonomous Decentralized Systems, Phoenix, AZ, 1995

[15] F. Cristian, S. Mishra, and G. Alvarez. ”High-performance asynchronous atomic broadcast”, Distributed Systems Engineering, Vol. 4, No. 2, Pages 109-128, June 1997.

[16] S. Floyd and V. Jacobson. ”Random Early Detection Gateways for Congestion Avoidance”, IEEE/ACM Transactions on Networks, Vol. 1, No. 4, Pages 397-413, August 1993.

[17] L. Gu. ”Towards Improving Atomic Broadcast Performance with Bandwidth Reservation”, MS Thesis, School of Electrical Engineering and Computer Science, Washington State University, December, 2000. URL: http://www.eecs.wsu.edu/ bakken/lgu-thesis.pdf

[18] M. Hayden. ”The Ensemble System”, Ph.D. Dissertation. Department of Computer Science, Cornell University, Ithaca, NY. Available as Tech. Rep. TR 98-1662, 1998

[19] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. ”Assured Forwarding PHB Group”, RFC 2597. June 1999.

[20] R. R. Koch, L. E. Moser, and P. M. Melliar-Smith. ”A Reliable Many-to-Many Multicast Protocol for Group Communication over ATM Networks”. The International Conference on Dependable Systems and Networks, Pages 337-346, New York, June 2000.

[21] X. Liu, R. van Renesse, M. Bickford, C. Kreitz, and R. Constable. ”Protocol Switching: Exploiting Meta-Properties”, submitted to International Workshop on Applied Reliable Group Communication, IEEE, Scottsdale, Arizona, April, 2001.

[22] S. Mishra and S. M. Kuntur. ”Improving Performance of Atomic Broadcast Protocols Using the Newsmonger Technique”, Dependable Computing for Critical Applications 7, January 1999.

[23] O. Ozkasap, Z. Xiao, and K. Birman. ”Scalabity of Two Reliable Multicast Protocols”, Department of Computer Science, Cornell University, Ithaca, NY. Available as Tech. Rep. TR 99- 1748, 1999.

[24] P. Pieda, J. Ethridge, M. Baines, and F. Shallwani. ”A Network Simulator Differentiated Services Implementation”, Open IP, Nortel Networks, http://www7.nortel.com:8080/CTL/#software, July 2000.

[25] M. Reiter and K. Birman. ”How to Securely Replicate Services”, ACM Transactions on Programming Languages and Systems, 16(3), May, 1994, 986-1009.

[26] Q. Sun, and D. Sturman. ”A Gossip-based Reliable Multicast for Large-scale High-throughput Applications”, The International Conference on Dependable Systems and Networks, Pages 347-358, New York, June 2000.

[27] R. Vitenberg, I. Keidar, and G. V. Chockler. ”Group Communication Specifications: A Comprehensive Study”, ACM Computing Surveys 33(4), pages 1-43, December 2001.

[28] X. Defago, A. Schiper and P. Urban. ”Totally Ordered Broadcast and Multicast Algorithms: A Comprehensive Survey”, Technical Report DSC/2000/036, Ecole Polytechnique Federale de Lausanne, Switzerland, September 2000.

[29] I. Keidar. ”Group Communication”, Chapter in Encyclopedia of Distributed Computing, J. Urban and P. Dasgupta, eds., Kluwer Academic Publishers, 2002, to appear.

[30] C. Williamson. ”Internet Traffic Measurement”, IEEE Internet Computing, December 2001.

[31] W. Willinger, M.S. Taqqu, R. Sherman, and D.V. Wilson, ”Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level”, IEEE/ACM Transactions on Networking 5(1) (Feb.1997), 71-86.