Message Buffering in Epidemic Data Dissemination[◊]

Emrah Ahi[*], Mine Çağlar[**] and Öznur Özkasap[***]
[*] Department. of Computational Science and Engineering
[**] Department. of Mathematics
[***] Department. of Computer Engineering
Koç University, Istanbul, Turkey
{eahi|mcaglar|oozkasap }@ku.edu.tr

***Abstract*—In reliable group communication, epidemic or probabilistic protocols gained popularity due to their scalability to large number of peers and robustness against network failures. Reliability properties of these protocols are ensured via probabilistic guarantees. A key issue to consider when offering reliability is the buffer space used by individual peers of the group. Our aim is to optimize the buffer space while providing reliability in epidemic data dissemination protocols. We introduce a novel randomized model and compare it with a hash-based approach for buffer management. The effect of short and long term buffering of peers and the buffer size on delivery latency and reliability are considered. We compute the performance measures through simulations of large-scale application scenarios.**

Keywords: Buffer Management, Epidemic, reliability

I. INTRODUCTION

Many distributed applications need dissemination of data from a single source to a large group of peers. Epidemic or gossip-based algorithms work with a probabilistic guarantee in an efficient way. Some application areas are failure detection [1], data aggregation, resource discovery and monitoring [2], and database replication [3]. In these protocols a single down or slow receiver cannot affect the whole system while the network load is distributed to all members. The gossiping mechanism provides a high resilience to problems like link failures or failure on a single node.

In an epidemic algorithm, every process of the system is potentially involved in the dissemination of messages [4]. Every process buffers every message (information unit) it receives up to a certain *buffer capacity b*, and forwards that message a limited number of *times t*. The process forwards the message each time to a randomly selected set of processes of limited size *f*, called the *fan-out* of the dissemination. The reliability of information delivery depends both on these values and the size of the system *n*. Through epidemic dissemination, eventually the message will be received by all members with high probability in the order of $\log n$ rounds. The probabilistic guarantee of message delivery in epidemic algorithms is directly related to the value of the dissemination parameters. These parameters can be tuned so that with arbitrarily high probability, the algorithm meets the guarantees that deterministic algorithms would provide.

In the simplest form of epidemic dissemination, if a member receives a message it directly forwards the message to a randomly chosen member. This scheme is fast but has a problem. A message can be sent to a node several times unnecessarily. The more sophisticated approach used in recent algorithms as well as this study is *anti-entropy* where the unnecessary transmission problem is handled by introducing a request-transmit procedure. Each member periodically chooses *f* random members from its neighbors and sends its message history. As a result, each peer detects the missing messages and requests from the other peers.

In this study, we are interested in the buffer management problem in epidemic information dissemination. To understand the buffering problem, recall the principle of a simple epidemic broadcast algorithm: every process that receives a message (information unit) has to buffer it up to a certain buffer capacity, and forward it a limited number of times, each time to a randomly selected set of processes of limited size (fan-out). Depending on the rate of new information production in the system, the buffer capacity may be insufficient to ensure that every message is buffered long enough so that it can be forwarded a sufficient number of times to achieve an acceptable reliability. Three complementary approaches have been considered in earlier work on buffer management. For reducing the memory usage, mechanisms for message drop and the number of nodes which buffer a given message are determined [5,6]. Another approach is network flow control where the idea is to influence the application by regulating its rate when processes do not have enough resources and enough time to buffer [7,8]. In message stability approach, the members inform the other peers in their view about which messages they buffer. If the members detect that all have received a specific message, they drop it from their buffers [9, 10].

Our contribution is a novel buffer management approach which we compare with the hash-based buffer management given in [5]. We set the long term

[◊] This work is supported in part by TUBITAK (The Scientific and Technical Research Council of Turkey) under CAREER Award Grant 104E064.

bufferers of a message at its origination instance. Then, the messages are disseminated epidemically through anti-entropy mechanism. Our model follows the FIFO scheme where in case of a buffer overflow the message that arrived earliest is dropped. In our scheme, not all nodes buffer all messages, instead a limited number of them buffer as many as their buffer space allows. Our preliminary study [11] with simple epidemics has shown that our approach is scalable and the load on the nodes is well distributed.

The paper is organized as follows. Section 2 summarizes the related work in message buffering. Section 3 introduces our approach. In Section 4, performance results of our scheme are examined. Finally, Section 5 concludes the paper.

II. RELATED WORK

In order to achieve reliability in group communication, the error recovery mechanism must be well designed. An efficient buffer management scheme is indispensable part of an error recovery mechanism.

Approaches for buffer management can be classified into three categories: optimizing the memory usage, flow control and providing message stability. In addition, different policies for the replacement of buffer items are considered.

A. Reducing the Memory Usage

The pioneering study [5] focuses on reducing the buffer requirement by buffering each message only over a small set of members. Upon receiving a message, a member determines whether it should buffer the message using a hash function based on its network address and the identifier of the message. However, dynamic redefinition of the hash table is not considered in this algorithm. In the present work, the long-term bufferers of a message are randomly chosen when it is created at the source. Hence, if a new member joins the system, it is eligible to be a bufferer as chosen by the source.

A multicast protocol that reduces buffer requirements is Randomized Reliable Multicast Protocol [6] which is an improvement over Bimodal Multicast [13]. Both protocols use epidemic error recovery. The message is kept in the long-term buffer for a fixed amount of time. In our approach, the messages are kept in the short-term and long-term buffers until the capacity of the buffers are reached.

B. Network Flow Control

Flow control is an adaptive mechanism that deals with varying resources such as CPU and bandwidth in the end hosts. A NAK based retransmission control scheme is given in [7]. The sender reduces its transmission rate whenever it receives too many NAKs from the receivers. It also keeps a log of its past transmission rates to prevent high decrease in the rate. This mechanism helps to minimize the buffer overflows at the receivers.

A different idea explored in [8] requires every process to calculate the average buffer capacity among all processes it communicates with and transmit that information. When the rate is too high with respect to the average, the process reduces the rate locally. Indirectly, the sources of the information get such a feedback and they reduce the rate of information production. The main drawback here is that the rate is adjusted according to the process with the smallest buffer space.

C. Achieving Stability

In [9], a stability detection algorithm is given for discarding the messages from the buffers of the peers. A message is said to be stable when it is delivered to all members of the group. All the members periodically exchange messages to inform each other about the messages they have received. When a member becomes aware of a message becoming stable, it safely discards the message. So the system wide buffer space is reduced. A drawback is the high traffic caused by frequent exchange of history messages.

Search Party [10] is another protocol in which contribution of a timer helps to discard packets from the buffers. All the members discard packets after a fixed amount of time to achieve stability.

Recently, a heuristic buffer management method using both ACKs and NAKs is proposed in [12] to provide scalability and reliability. In every group of receivers there are one or more members with higher error rates than the other members. These nodes are the ones with the least reliable nodes and the ones with the slowest links. The idea is that if a message is correctly received by these nodes, it must have been received by the other nodes. The nodes with the most reliable links are selected to buffer the received messages for a fixed amount of time.

D. Replacement Policy for Buffer Items

Network Friendly Epidemic Multicast [13] combines a standard epidemic protocol with some complementary mechanisms. A novel buffering technique that combines different selection techniques is proposed to discard messages in case of a buffer overflow. During the network congestion periods these selection mechanisms help to control the flow of the network. The used selection strategies for discarding messages are random purging, age-based purging and semantic purging. Random purging is to discard an item from buffer randomly. It can be helpful when the system is congested for some time. Age-based purging is simply discarding the oldest message. And semantic purging means that a message which has been recognized as obsolete is discarded. Obsolescence relation is determined by the application.

Recently, least recently used (LRU) buffer replacement scheme is considered in [14] for epidemic information dissemination. The buffer hit rate of the system, defined as the probability of retrieving requested item matching the key of request either from the local buffer, the origin device, or the buffer of a remote device, is also determined. In LRU scheme, a new coming message is placed on the first position and the message at the rear is discarded as in our case. However, when a request arrives for a message in the buffer, that message is placed into the first place by moving the items in front one position down. Hence, the least used item stays at the rear of the stack possibly next to be discarded.

III. OUR MODEL

We propose an efficient buffering algorithm that ensures reliability in epidemic information dissemination, adaptable to dynamic groups and reduces the system wide buffer space. In our scheme, every node has a short-term and a long-term buffer for retransmissions. Message discarding policy is not time dependent like [10]. We use FIFO ordering for messages in the buffer. A new coming message is placed on the first position in the buffer stack. The oldest message in the buffer which is at the rear of the stack is discarded in case the capacity of the buffer is reached. Another aim of the algorithm is to reduce the number of bufferers in the system, but it has no use of a hash function like [5] to determine the bufferers. Typical applications include those where messages originate from a single source and all data is to be distributed to all peers. Our aim is to disseminate the data as quickly as possible and efficiently by uniform buffer usage.

A. Determining Bufferers

When a message is generated, a set of long-term bufferers for the message is determined by the source and the ids of these bufferers are piggybacked to the message. The message is directly forwarded to bufferer processes. The bufferer nodes for a message are determined randomly from the peers known to the message source. In addition, the number of bufferers must be chosen large enough to handle failures in some bufferers and small enough not to give rise to overhead in the traffic. The number of bufferers is an important parameter for the system efficiency. The bufferer processes may hold the corresponding messages in their long-term buffers for ever if there is space. If a process detects that it has missed a message, it can request the message from one of the long-term bufferers of that message. The aim is to keep the probability of a message being removed from the long-term buffers of all bufferers very small when there are members missing that message.

B. Dissemination

The messages are disseminated to all members epidemically by the anti-entropy model. Periodically, all members choose f peers randomly and then send the information of the messages received up to that time. This process is called gossiping and the parameter f is the fan-out. This history information of received messages is called a digest message. In our scheme, the digest message also contains the ids of the long-term bufferers and the information that whether the node that sent the digest message has discarded the corresponding message or it has the message in the short-term buffer. Relying on this information, the node that has received the digest message requests the data from the source of the digest or from one of the long-term bufferers. The short-term buffer is preferred. If the owner of the digest cannot serve the request from its short-term buffer, then the requester can ask one of the long-term bufferers for the missing message. The aim is to distribute the load of buffering over the network. If the long-term bufferer fails to retransmit the message, the request can be forwarded to another bufferer.

When a member receives a new message, it takes the message to its short-term buffer. If the short-term buffer is full, the oldest message is removed.

Fig.s 1 and 2 illustrate our idea with a simple scenario. The columns next to the nodes represent the long-term and the short-term buffers of the members respectively. The list written in curly braces is the message history, that is, the messages received up to that time by the node. There are 6 messages sent to the group.

In Fig. 1, node 4 gossips to 2 and node 3 to 1. When node 2 gets the digest message of node 4, it realizes that it has not received message 1 which node 4 received. Then it requests message 1 from node 4, but since node 4 dropped message 1 from its short-term buffer it cannot handle that request. Then, since the digest message contains the bufferers of the messages, node 2 requests the message from the bufferer of message 1 which is node 3 as shown in Fig. 2. Similarly, node 1 also detects that it missed messages 3 and 5. It gets message 5 from node 3, but cannot retrieve message 3 which it requests from the bufferer, namely node 2.

C. Algorithm for a Node

The point of view for a node in the system is given by the following algorithm.

```
For all gossip rounds
    Choose random f destinations
    Send digest msg to each destination
Event digest arrives
    Compare msg history with digest
    Determine missing messages
    If missing msg is contained in source of digest
        Send request msg to source of digest
    Else if the source of digest does not contain the msg
        Send request to one of the bufferers
Event request arrives
    If short term or long term buffer contains data
        Send requested data to request source
Event data arrives
    Add data to buffer
    Add data information to msg history
If the node is data source
    Create data msg
    Choose bufferers for that message randomly
    Send the message to the bufferers
```

IV. SIMULATION

A. Simulation Model

The simulation is implemented in Java where all the nodes in the system can perform their tasks simultaneously because each node is individually a thread. In our simulations, we use a system with 100 peers where we assume that all members have the information of full membership. All the members have links to all other members. In other words, we have a fully connected graph. The links are symmetric and identical. Each link allocates a bandwidth and there is a configurable latency for sending a message from one node to another. Our aim is to disseminate 1000

messages. At every 10 milliseconds the source generates a message and immediately sends it to the bufferers. At every 100 milliseconds, all the members do gossiping. The fan-out parameter is set to 5.

The size of the data messages is assumed to be 1024 bytes. The size of the request messages is 1 byte. Also the size of the digest messages is assumed as *k* bytes where *k* is the number of entries in the digest message.

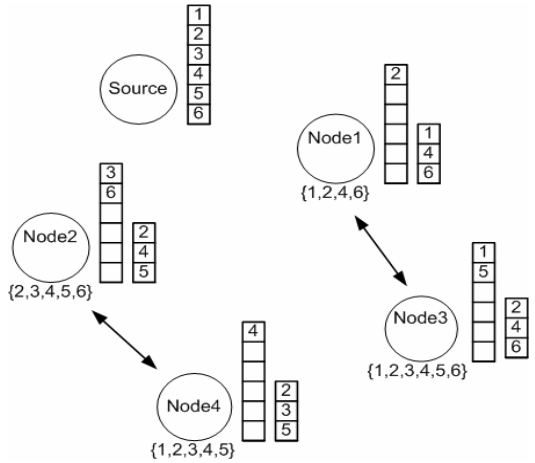

Figure 1. Gossiping

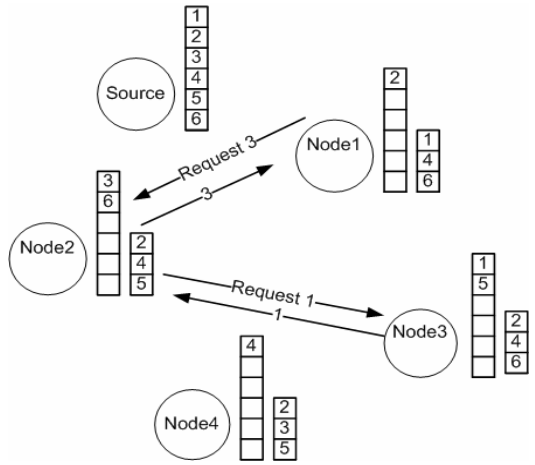

Figure 2. Requesting missed messages from bufferers

Links in the network is bidirectional and each direction has a bandwidth of 1 M bps. The propagation delay on a link is assumed as *p*=5 milliseconds. So, the time needed to send a packet from a node to another is $t = p + s/b$ where *s* is the number of bytes the message has and *b* is the bandwidth. So, time needed to send a data message is $5+8000/10^6$ millisecond. Therefore, the time needed to send a request message is approximately 5 msec. The time for digest messages is variable according to the number of messages that the node received. The size of the digest message can vary from 1 to 1000 bytes. So the delay for a digest message ranges from 5 msec to 13 msec. On the other hand, the data messages take 13 msec to forward. The queuing delays and the processing delays are ignored during simulations.

B. Simulation Parameters

Fan-out is the number of nodes that are chosen for sending the digest message during a gossip round. *Gossip-round* is the time period of gossiping. *Long-term buffer size* denotes the maximum number of messages that the long-term buffer of a node can hold and *short-term buffer size* denotes the maximum number of messages that the short-term buffer of a node can hold. *Message origination rate* is the number of messages generated by the source node in one sec. *Average used long buffer space* is the mean number of messages in the long term buffer of all the nodes in the system. *Average used short buffer space* is the mean number of messages in the short term buffer of all the nodes in the system. *Average long buffering time* is the mean time that a message spends in the long term-buffer of a node is the *average short buffering time* is the time that a message spends in the short term-buffer of a node. *Message dissemination time* is the time that passes until a node receives all generated messages. *Message receiving time* denotes the time that passes from origination of a message to its reception by a node. Lastly, *Percentage of Short-Term* is the percentage of the received messages that are sent from short-term buffers of the nodes.

C. Simulation Results

In this section, we compare performance results of our model and the protocol where the hash-based buffering approach described in [5] is used for epidemic information dissemination. Originally in [5], the messages are multicast and then the error recovery is performed. A node finds the bufferers of the missed messages by the use of a hash function and sends requests. The values for parameters are 50 msg for long-term buffer size and 20 for short-term buffer size.

To compare our model with the hash based approach, we consider the situation where the number of messages kept in the long-term buffer of a node is approximately equal in both cases, in particular about 80. So, the number of long-term bufferers in our model is set to 8 as 1000 messages are received by 100 members to have 8*1000/100=80. On the other hand, the parameters of the hash function are adjusted accordingly to have approximately 80 different messages passing through each long-term buffer in the hash based approach.

In Fig. 3, we examine the dissemination time, namely the time needed to send all 1000 messages to all 100 peers, in each approach. We see that the dissemination of all data is completed in the hash-based approach approximately 1.5 sec later than our model. In our model, the bufferers are determined at the generation of the message and the messages are directly sent to the bufferers. However, in the hash-based approach a node decides to be a bufferer for a message when it receives the message through gossiping eventually.

Comparison of the mean time that a message spends in the long term buffer of a node in both approaches is given in Fig. 4. We can see that the time spent in the long term buffer in our approach is 0.1 sec (100 msec) higher than the hash-based one. This means that in our model a node serves a message for a longer time. So, during dissemination the availability of a message is more likely in our model. This affects also the whole dissemination time given in Fig. 3.

Similarly, the time for a message that a node holds in its short-term buffer is higher in our approach. This can

be observed from Fig. 5. The mean short-term buffering time is 0.61 sec in our random model and 0.54 sec in the hash-based one.

The mean time that passes from origination to receiving of a message is another important measure in dissemination of data. If a node receives a message earlier, this means that the corresponding message can be reached from the other nodes earlier for repair. So, this will make an important effect on the dissemination time. Message receiving times in both approaches are given in Fig. 6. The mean receiving time is 1.6 second in our approach and 2.4 second in the hash-based one.

The utilization of the bufferers is another performance measure that we consider. In our model, the load imposed on the bufferers of a message and the nodes holding the corresponding message is equal. From Fig. 7, we can observe that approximately 500 of the messages are received from short-term buffers of the nodes holding the messages. The remaining 500 messages are received from the bufferers of the messages. On the other hand in the hash-based approach, only %10 of messages is received from the bufferers of the corresponding messages. The remaining 90% comes from the short-term buffers of the peers. In our approach the messages are directly forwarded to the bufferers. So, during a gossip round if a member fails to receive a message, it can request the message from one of the bufferers. On the other hand, in the hash-based approach, it is likely that the bufferers have not received the message during earlier times of the dissemination. This is due to the fact that even the long-term bufferers of a message obtain it through the gossip and request mechanism.

Another measure that must be taken into account is reliability. In Fig. 8, the effect of reliability in both approaches can be examined. We can clearly observe that the short-term buffer size is not an important factor that affects the reliability in our approach. But the hash-based approach is very sensitive to the size of the short-term buffer due to the reasons explained for Fig. 8. For example, if the short-term buffer size is 5, about 20% of the messages are lost. If we increase it 3 folds, the number of missed messages decreases by 10 folds.

In Fig. 9, we investigate the effect of the long-term buffer size. We increase the long-term buffer size from 2 to 30 in both approaches. In our model after the long-term buffer size is 10 messages, the dissemination time becomes almost constant which is 10.2 sec. In the hash-based approach, reliability is obtained when the long-term buffer size is greater than 4 messages. The dissemination time decreases steadily, but remains higher than our model.

Up to this point in the simulations, we have investigated performance measures with the number of nodes equal to 100. In Fig. 10, we examine the number of different messages stored in the long-term buffer per peer while the size of the group increases. Note that the long-term buffer space is 50 for all sizes.

We implemented LRU buffering scheme in our simulations as well and repeated all the analysis above. However, we could not observe any significant effect on the performance measures.

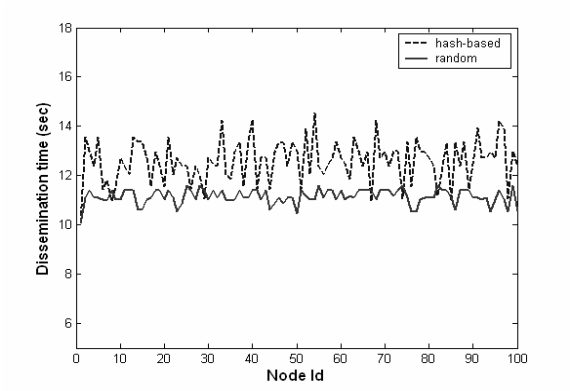

Figure 3. Comparison of dissemination times
Short-term size: 20 Long-term size: 50

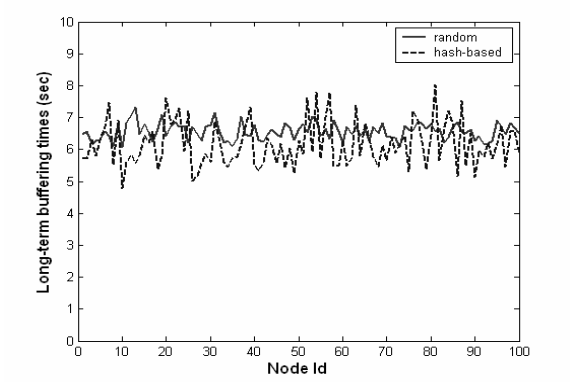

Figure 4. Comparison of long-term buffering times
Short-term size: 20 Long-term size: 50

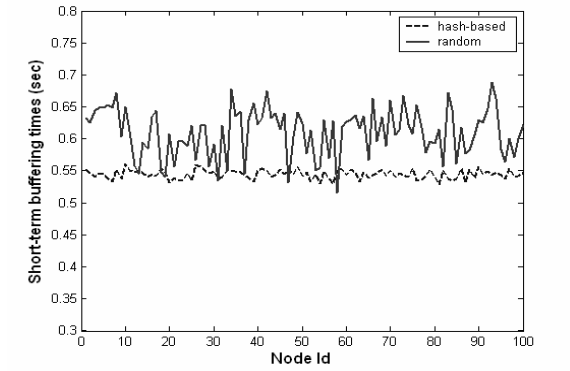

Figure 5: Comparison of short-term buffering times
Short-term size: 20 Long-term size: 50

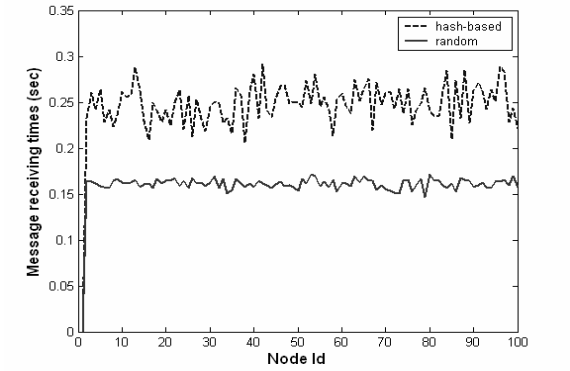

Figure 6. Comparison of message receiving times
Short-term size: 20 Long-term size: 50

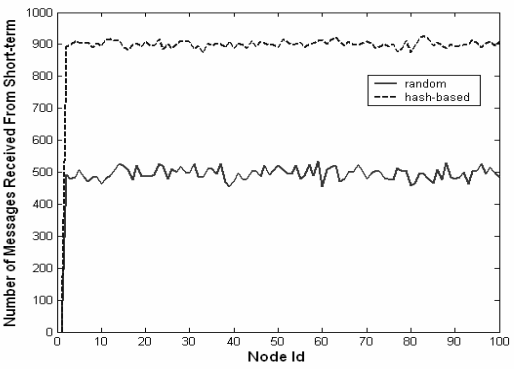

Figure 7. Number of messages received from short term
Short-term size: 20 Long-term size: 50

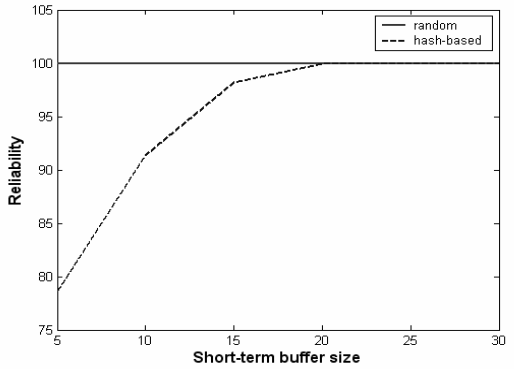

Figure 8. Short-term buffer size versus reliability
Long-term size: 50

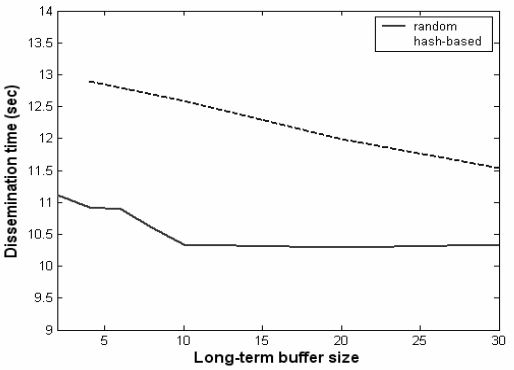

Figure 9. Long-term buffer size versus dissemination time
Short-term size: 20

V. CONCLUSION

We have proposed a buffer management scheme applicable to dissemination of data to a large group of nodes where epidemic dissemination idea is used. The results have shown that our approach reduces the dissemination time and the receiving time of all data, increases the utilization of buffers and reliability of dissemination. As future work, we will work on an analytical model of our approach.

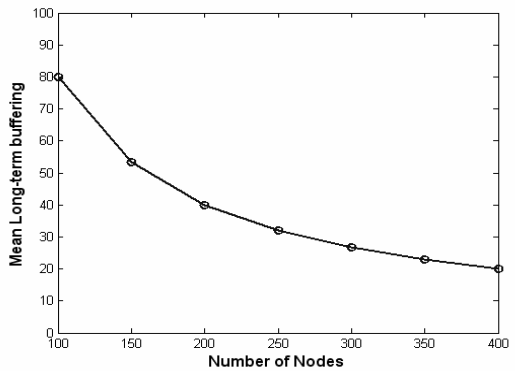

Figure 10. Number of nodes – Mean long-term buffering

REFERENCES

[1] R. van Renesse, Y. Minsky, and M. Hayden, "A Gossip-Style Failure Detection Service," *Middleware 98: IFIP Int'l Conf. Distributed Systems and Platforms and Open Distributed Processing,* N. Davies, K. Raymond, and J. Seitz, eds., Springer, 1998, pp. 55-70.

[2] R. van Renesse, K.P. Birman, and W. Vogels, "Astrolabe: A Robust and Scalable Technology for Distributed Systems Monitoring, Management, and Data Mining," *ACM Trans. Computer Systems,* vol. 21, no. 2, 2003, pp. 164-206.

[3] A.J. Demers et al., "Epidemic Algorithms for Replicated Database Maintenance," *Proc. 6th Ann. ACM Symp. Principles of Distributed Computing,* ACM Press, 1987, pp. 1-12.

[4] Patrick T. Eugster, Rachid Guerraoui, Anne-Marie Kermarrec, Laurent Massoulie, "Epidemic information dissemination in Distributed Systems", *IEEE*,Computer, May 2004.

[5] O. Ozkasap, R. van Renesse, K.P. Birman, and Z. Xiao, "Efficient Buffering in Reliable Multicast Protocols," *Proc. of the First Int'l Workshop on Networked Group Communication* (*NGC' 99*), Pisa, Italy, Nov. 1999, pp. 188-203.

[6] Z. Xiao, K.P. Birman, and R. Renesse, "Optimizing Buffer Management for Reliable Multicast," *Proc. of the Int'l Conf. on Dependable Systems and Networks* (*DSN'02*), Washington, D.C.

[7] Yamamoto, M. Yamamoto, and H. Ikeda, "Performance Evaluation of ACK-Based and NAK-Based Flow Control Mechanisms for Reliable Multicast Comm.," *IEICE Trans. on Comm.*, vol. E84-B, no. 8, Aug. 2001, pp. 2313-2316K.

[8] L. Rodrigues, S. Handurukande, J. Orlando, R. Guerraoui, and A.-M. Kermarrec. "Adaptive gossip-based broadcast". In *IEEE International Conference on Dependable Systems and Networks (DSN)*, 2003.

[9] K. Guo and I. Rhee, "Message Stability Detection for Reliable Multicast," *Proc. of the 19th IEEE Conf. on Computer Comm.* (*INFOCOM 2000*), New York, USA, Mar. 2000, pp. 814-823.

[10] M. Costello and S. McCanne, "Search Party: Using Randomcast for Reliable Multicast with Local Recovery," *Proc. of the 18th IEEE Conf. on Computer Comm.* (*INFOCOM '99*), New York, USA, Mar. 1999, pp. 1256-1264.

[11] A. Alagöz, E. Ahi., O. Ozkasap, "Network Awareness and Buffer Management in Epidemic Information Dissemination" (poster paper), ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2005), July, 2005, Las Vegas

[12] Jean François Paris, Jinsun Baek, "A Heuristic Buffer Management and Retransmission Control Scheme for Tree-Based Reliable Multicast" ETRI Journal, Volume 27, Number 1, February 2005.

[13] J. Pereira, L. Rodrigues, M. Monteiro, R. Oliviera, A. M. Kermarrec, "Network Friendly Epidemic Multicast", *22nd International Symposium on Reliable Distributed Systems,* 2003 IEEE.

[14] C. Lindemann and O. Waldhorst. "Modelling Epidemic Information Dissemination on Mobile Devices with Finite Buffers.",*SIGMETRICS'05.*