Stepwise Probabilistic Buffering for Epidemic Information Dissemination

Emrah Ahi[*], Mine Çağlar[**] and Öznur Özkasap[***]
[*] Computational Sciences and Engineering
[**] Department of Mathematics
[***]Department of Computer Engineering

Koç University, Istanbul, Turkey
{eahi| mcaglar|oozkasap }@ku.edu.tr

Abstract— **For large-scale peer-to-peer applications, bio-inspired epidemic protocols have considerable advantages as they are robust against network failures, scalable and provide probabilistic reliability guarantees. While providing reliability, a key issue to consider is the usage of system wide buffer space. In this context, we introduce a novel scheme called stepwise probabilistic buffering that reduces the amount of buffering and distributes the load of buffering to the entire system where every peer does not have the complete view of the system. We compute the performance measures through simulations of large-scale application scenarios.**

Keywords-Buffering, epidemic, peer-to-peer, information dissemination, reliability, topology-aware.

I. Introduction

Peer-to-peer (P2P) communication in large scale settings has many applications in today's Internet and in these communication systems there is a need for a source to disseminate data to a large group of peers. Besides, a P2P dissemination system must be reliable, scalable and must provide a management of membership. Relying on these communication paradigms, epidemic or probabilistic protocols [1], [2] have significant advantages. They are simple to implement, inexpensive to run, robust and they impose a constant load on the links and receivers. The gossiping mechanism that is used to disseminate the data provides a high resilience to network problems like link failures, slow links or a failure on a single node. A significant issue is that these features of epidemic protocols are preserved as the scale of the system increases. However, during deployment of these protocols, real systems always have a limited capacity. Peers can exchange only the data messages they have buffered. Therefore, an efficient buffer management mechanism is a crucial issue in providing reliability for these protocols. Studies accomplished in this area emphasize several aspects of buffer management such as reducing memory usage, packet discarding policy and message stability.

Our contribution in this area is a novel buffer management model that reduces the memory usage of the system and distributes the load of buffering evenly to the entire system where all peers have only partial knowledge of the participants. In this model, only a small subset of the peer population keeps a data message in its long-term buffer so that buffering load on each peer does not increase as the system size increases. The long-term bufferers are determined through a stepwise search algorithm which is inspired by the random forwarding encountered in epidemic algorithms. The application area is P2P epidemic information dissemination where every peer has only a partial view of the system. Bufferer determination procedure is the novel part of our proposal which takes place concurrently with epidemic data dissemination. The major aim is to distribute the buffering load to the entire system evenly. We show that our Stepwise Probabilistic Buffering system facilitates the achievement of full reliability and faster data dissemination than a benchmark approach based on a hash function used for determining the bufferer nodes [3].

The rest of the paper is organized as follows. Section 2 gives an overview of the related work in buffer management. In Section 3, we describe our approach in detail. In Section 4, its performance is evaluated through simulations. Finally, Section 5 concludes the paper and describes the future work.

II. Related Work and Discussion

In order to achieve reliability in group communication, the error recovery mechanism must be well designed. An efficient buffer management scheme is an indispensable part of an error recovery mechanism. The existing approaches are designed for various aspects of buffer management, namely, flow control, optimization of the memory usage, providing message stability and the replacement of buffer items. In this section, we review the related work and compare with our approach.

A. Network Flow Control

Flow control is an adaptive mechanism that deals with varying resources such as CPU and bandwidth in the end hosts. In the NAK based retransmission control scheme given in [4], the sender reduces its transmission rate whenever it receives too many NAKs from the receivers. This mechanism helps to minimize the buffer overflows at the receivers.

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

A different idea explored in [5] 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. On the other hand, a source node reduces the rate of information production according to the process with the smallest buffer space.

B. Reducing the MemoryUsage

The pioneering study [3] 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. The hash function is devised so that the bufferers are chosen uniformly among the peers. However, when a new member joins the system it cannot become a bufferer as dynamic redefinition of the hash table is not considered.

In the present study, the messages are buffered by only a limited number of peers as well. The bufferers are selected through an adaptive scheme in order to distribute the buffering load uniformly. As a result, if a new member joins the system, it is eligible to be a bufferer.

A multicast protocol that reduces buffer requirements is Randomized Reliable Multicast Protocol [6] which uses epidemic error recovery. A message is kept in the long-term buffer for a fixed amount of time. In our approach, the messages remain in the buffers until the capacity of a buffer is reached.

Structured peer to peer networks such as Chord [7], CAN [8] and Tapestry [9] offer a management on participating peers and published data items. Chord is based on a ring, in Pastry and Tapestry hypercube is used, Tornado uses a tree structure. These systems name the participating peers and available data items with a distributed hash function. Chord [7] assigns keys to nodes with *consistent hashing*. With a high probability this function balances the load imposed on peers namely all nodes receive approximately the same amount of keys. Chord peers store a small amount of data and require partial membership information. A node resolves the hash function by communicating with other nodes because the hash function is distributed.

Another buffer management scheme which reduces memory usage is [10] where the members are organized as regions. In every region, the nodes with the most reliable links are responsible for buffering the data.

C. Achieving Stability

A message is said to be stable when it is delivered to all members of the group. There are buffer management approaches which explicitly take stability into account. In [11], all 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 [12] 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.

A heuristic buffer management method based on both ACKs and NAKs is proposed in [10] 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 and slowest links. The idea is that if a message is correctly received by these nodes, it has been probably received by all other nodes. In that case, the repair nodes that buffer the message can discard it.

Our protocol adjusts several parameters such as the number of bufferers and the buffer size to achieve stability with a high probability.

D. Replacement Policy for Buffer Items

Network Friendly Epidemic Multicast [13] combines a standard epidemic protocol with a novel buffering technique that combines different selection techniques for discarding messages in case of a buffer overflow. The used selection strategies are random purging, age-based purging and semantic purging. Random purging refers to discarding an item from the buffer randomly. 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.

Least recently used (LRU) buffer replacement scheme is considered in [14] for epidemic information dissemination. 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.

In this paper, a first-in-first-out policy equivalent to age-based purging is implemented in the case of a buffer overflow. We have run simulations also with LRU scheme and found no significant difference.

III. STEPWISE PROBABILISTIC BUFFERING

In this section, we describe the system model, epidemic approach used for data dissemination, overlay topology, bufferer determination procedure and optimizations incorporated to our approach for buffering.

A. System Model and Epidemic Dissemination

The system consists of peers connected through an overlay reflecting the properties of the underlying network topology. Each peer has a *partial* view of the system which is a quite plausible assumption considering a large scale distributed application scenario. A major aim of our approach is to be able to choose bufferers uniformly through the system so that the load of buffering would be well balanced among participating peers and the efficiency of content dissemination would be improved as a result. The approach also reduces the buffer usage since only a small subset of the peers is chosen as bufferers for each message. Furthermore, it is applicable to large scale scenarios, provides reliable delivery and is adaptable to dynamic join and leaves to the system.

A popular distribution model based on the theory of epidemics is the anti-entropy [15]. According to the terminology of epidemiology, a peer holding information or an update it is willing to share is called infectious. A peer is called susceptible if it has not yet received an update. In the anti-entropy process, non-faulty peers are always either susceptible or infectious. In this model, periodically, each peer picks *f* (fan-out) other peers at random, and exchanges its state information with the selected one. For spreading information, our system uses a pull-based approach in which data dissemination is triggered by susceptible peers when they are picked as gossip destinations by infectious peers.

In our model, every message is originated from a source peer. In this protocol, each peer periodically selects *f* random peers from its partial view and sends them a digest including its recent message history. Digest of a peer contains the state information for the last *d* messages the peer has received so far and identifiers of their bufferers. Upon receiving a digest, a peer may determine the messages that it lacks and can request them from the bufferers indicated in the digest for retransmission. If a bufferer has crashed or cannot retransmit the message, the request can be forwarded to another bufferer.

Each peer has a short-term and long-term buffer. Once a data message is received by a peer, it is kept in its limited short-term buffer until it becomes old enough to discard. The short-term buffer is useful during epidemic dissemination intervals. On the other hand, when a peer becomes bufferer for a particular data, the data is kept in its long-term buffer. The long-term buffer is useful for achieving reliability in data dissemination. For both short and long-term buffers, either FIFO or LRU drop policy is employed. Message discarding policy is not time dependent like in [11].

B. *Overlay Topology*

We assume the existence of an overlay among peers reflecting the properties of the underlying network topology, and consider a transit-stub model as a good approximation of the Internet topology. The Internet can be viewed as a set of interconnected routing domains where each domain can be classified as either a stub or a transit domain. Stub domains correspond to interconnected local area networks and the transit domains model wide or metropolitan area networks. A transit domain is composed of backbone nodes which are well connected to each other with high bandwidth links. Every transit node is connected to one or more stub domains.

C. *Bufferer Determination Procedure*

The process of determining the bufferers of a data message is initiated by the source. When the bufferers are determined their ids are piggybacked to the data message and sent to the bufferers firstly. Bufferer determination procedure is the novel part of our proposal which takes place concurrently with epidemic data dissemination. The major aim is to distribute the buffering load to the entire system evenly. As bufferers are distributed evenly among the peers, the load of cooperative data dissemination would also be well distributed among the peers.

For determining the bufferers of a data message, the source sends buffering request messages to randomly selected *b* peers in its partial view. Parameter *b* is the number of bufferers per message. For a data message, if *b* > 1 then its bufferers are determined in parallel. *Buffer fullness ratio of a peer* (*BF*) is the ratio of the number of messages that are stored in the peer's buffer to its long-term buffer capacity. *Steps-to-Live* (*STL*) value attached to a buffering request message indicates the maximum number of times that request message can be forwarded among peers. When a peer receives a buffering request message for a particular data, it accepts the request with probability (1 – *BF).* Otherwise, it forwards the message to a randomly selected peer from its partial view with a probability equal to *BF*. For example, if 90% of the long-term buffer is full, then the peer becomes the bufferer of the message with probability of 0.1 and sends the buffering request to one of its neighbors with probability of 0.9. Fig. 1 shows the steps of bufferer selection mechanism. Initially, assuming that all buffers are empty, peers that are in the partial view of the source will accept the buffering requests with higher probabilities. Then, as the buffer level of these neighboring peers will approach their capacity, they will begin to forward the buffering requests with higher probabilities to their neighboring nodes. Likewise, as the data dissemination continues, the peers with one or more hops away from the source will begin to reach their buffer capacities and forward the buffering requests to their neighbors. Thus, a stepwise probabilistic buffering takes place. When a peer becomes bufferer of a message it announces that back to the source. When the entire bufferer announcement messages return to the source, the source includes the ids of these bufferers in the data, sends data to the bufferers firstly, and the epidemic data dissemination takes place.

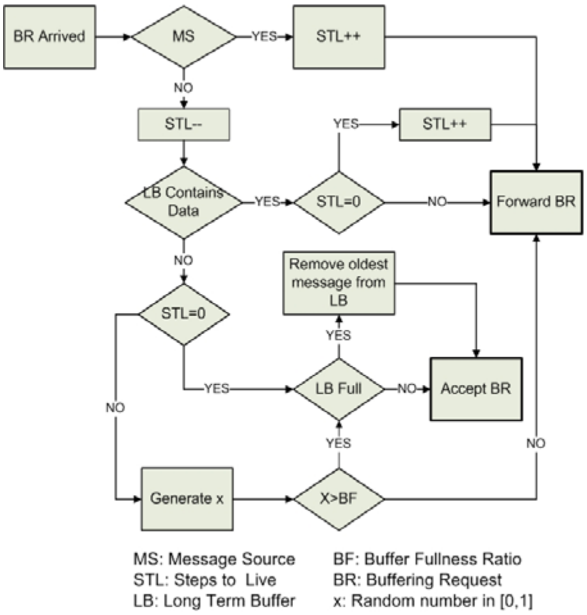

Figure 1: Flow chart for determining bufferers

D. *Optimizations*

There is a trade-off in the decision for the STL value of bufferer request messages. If the STL value is chosen large enough, uniform selection of bufferers would be easily achieved since the request message will be able to visit more peers in the overlay and find a suitable buffer place for itself. However, there is the cost of higher delays caused due to the bufferer determination rounds. In order to provide uniform selection of bufferers, we integrate the following optimizations to our approach.

1) *Last Forwarders:*

In this optimization, the ids of the last *n* forwarders are included in the buffering request messages. Via this information, a bufferer request is not resent to the last n forwarders and the STL mechanism is used more efficiently. The idea is that the peers that have forwarded the request have probably approached their buffer capacities. Therefore, resending the buffering request to such a peer is a redundant task.

Fig. 2 illustrates the mechanism with a simple example for the case $n = 3$. Assume that the partial views of the peers include one hop neighbors. P1 invokes the buffering mechanism for a particular data, writes its id to the buffering request and forwards it to P2. Similarly, P2 writes its id to the buffering request and sends it to P3. Next, P3 does the same process and forwards the request to P4. Since P2 and P3 are in the last forwarders list, P4 does not send the buffering request to these nodes and sends it to P5. In the same way, P5 does not send the request to P3 or P4, but to P6.

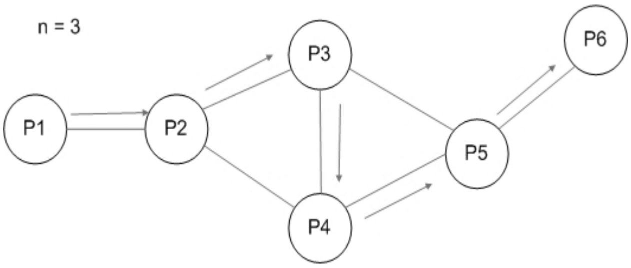

Figure 2: Illustration of last forwarders mechanism

2) Considering Overlay Topology:
This optimization is also incorporated for providing uniform bufferer selection. The idea is to assign different forwarding probabilities to peers according to their topological properties on the overlay. Therefore, this mechanism provides topology-awareness. Forwarding probability of a peer corresponds to the probability of sending a buffering request to a neighbor node. According to the types of neighbor nodes, the probability of forwarding a buffering request at a peer would differ.

We define three types of nodes according to their location on the transit-stub overlay, namely *transit (T), intermediate (I)* and *stub (S)*. An intermediate node connects a transit node to a stub domain. For example, the nodes labeled *r* and *t* in Fig. 3 are intermediate nodes. Two transit nodes are connected by high-delay intra-transit links (*TT*). A transit and an intermediate link are connected via intermediate delay stub-transit links (*TI, and IT*). Likewise, there exist low delay intra-stub (*SS, IS, and SI*) links in stub domains of the overlay. In our model, we assign forwarding probabilities P_{xy} to peers according to their topological properties as follows:

For a *T* node: $P_{TT} > P_{TI}$

For an *I* node: $P_{IT} > P_{IS}$

For a *S* node: $P_{SI} > P_{SS}$

As an example, assume that node *p* in Fig. 3 is the message source. If the transit source *p* sends the buffering request with equal probabilities to the nodes in its partial view (i.e. one-hop neighbors), then the nodes in stub-domains 3, 4 and 5 will accept the buffering requests less than the nodes in the stub-domains 1 and 2 that are directly connected to the source. Considering topology-awareness, if the source is a transit node then we assign a higher probability of forwarding the request to transit neighbors than forwarding the request to stub neighbors.

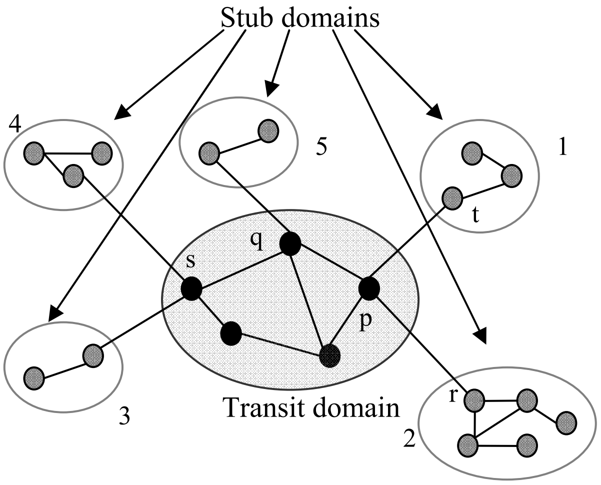

Figure 3: Overlay topology and forwarding probabilities

Non-source transit nodes also send the request to their transit neighbors with a higher probability. An intermediate node sends the request to the transit node with a higher probability than its other neighbors, namely stub nodes. A node in stub-domain forwards the request to one of its neighbors with equal probabilities. In Fig. 3, when node q receives the buffering request from node p, it sends the request to node s with a higher probability than sending it to its neighbor in stub-domain 5. If node t receives a buffering request from a node in its stub domain, then it sends the request to p with a higher probability.

IV. Simulation Results

The simulation is implemented in Java where a discrete time event based model is used. For the network topology, we use the gt-itm internetwork topology modeling tool [16] for generating transit-stub topologies. The optimizations mentioned in the previous section are integrated to the simulations and last forwarders value is set to 20.

A. Uniform Bufferer Selection

In the first part of the experiments, we investigate how well Stepwise Probabilistic Buffering achieves uniform bufferer selection on a controlled topology. The messages are generated from a single source where the total number of messages generated is equal to the total long-term buffer capacity of the system. Our aim is to observe whether the messages are distributed evenly to the long-term buffers or not. There are 100 nodes in the system where 4 of them are transit and every transit node is connected to 2 stub domains on the average. The mean number of nodes in each stub-domain is 12. The transit nodes are connected to each other with the probability of 0.8 and each node in a stub-domain is connected to another node in its domain with the probability of 0.5. Fig. 4 shows the sketch of the topology where node numbers are indicated. The long-term buffer capacity of a node is

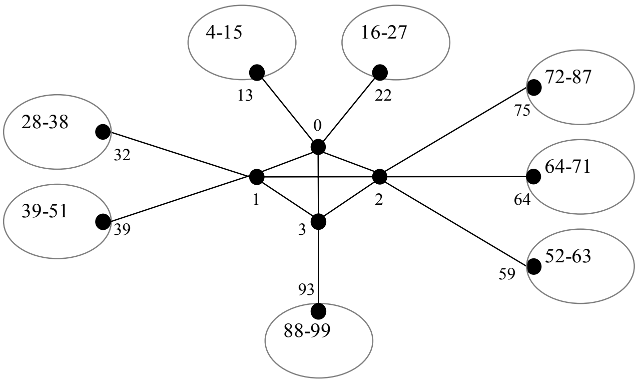

Figure 4: Simulation Topology

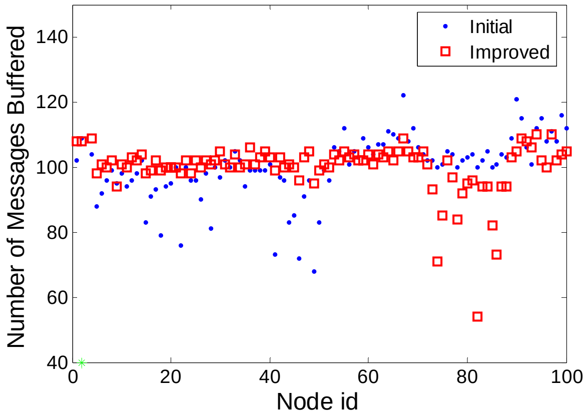

Figure 5: Source is a transit node
Std. initial = 13.97, Std. improved = 12.72

100 messages. We let the source node generate 10,000 messages just equal to the capacity of all long term buffers in a network of 100 nodes. The performance metrics for these set of simulations are as follows:

- *Retention ratio* is the ratio of the number of messages retained in all long-term buffers to the total number of messages generated.
- *Scattering ratio* is the ratio of the number of distinct sub-domains a message is buffered to the total number of bufferers of the message.

In Figures 5 through 7, the source node is varied in terms of its position in the overlay. Fig. 5 shows the total number of messages buffered by each peer when a transit node, id 2, is chosen to be the message source. Initial forwarding probability values and improvements over these values are used to obtain the two set of buffer levels. The standard deviation of the messages buffered among all nodes is given as a distinguishing metric for comparison of uniformity over all buffers. In our experiments, when a forwarding probability distribution leads to buffering of larger number of messages in certain domains such as those close to the source, the probability of bouncing back to the transit nodes from those domains is increased. Several trials have yielded a more uniform buffer load.

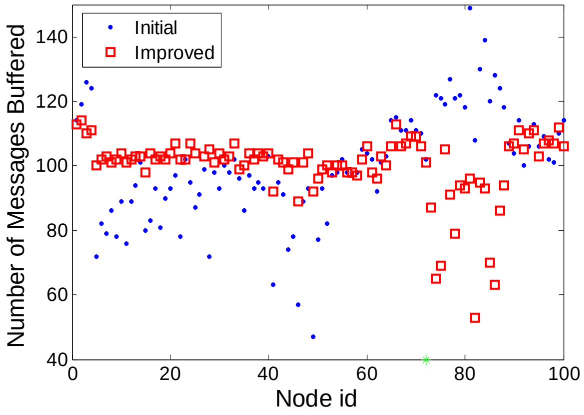

Figure 6: Source is a stub node
Std. initial = 19.78, Std. improved = 14.34

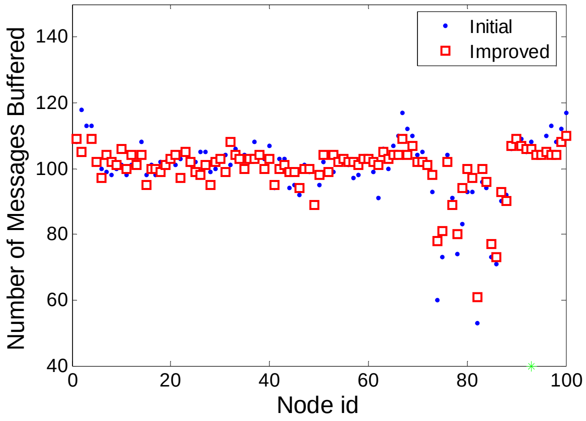

Figure 7: Source is a stub node that has a transit neighbor
Std. initial = 14.52, Std. improved = 12.64

As a stub node, node 72 is chosen to be the source in Fig. 6. In this case, the variance is somewhat higher than the transit source. This can be explained by the larger variation in the number of peers connected in a stub domain. In Fig. 7, the message source is an intermediate node, node 93, which is directly attached to a transit node. The uniformity of the buffer load distribution is close to that in Fig. 5 where the source is a transit node. In all cases, some nodes belonging to the domain of the nodes 72-87 buffer fewer messages. The reason for this is the relatively higher number of nodes in this domain, namely 16, compared to the expected value 12. Besides, it is connected to transit node 2 which has 3 stub domains, a higher number than the average number 2. As a result, the buffering requests reach to this domain less frequently.

We investigate the transient behavior of the uniformity of buffer fullness by monitoring the standard deviation of the buffer levels as the message generation proceeds. For this purpose, the standard deviation scaled by the mean of the used buffer space of all nodes is plotted against the proportion of messages generated in Fig. 8. When the generated messages approach the full long-term buffer capacity of the system, the variability decreases which indicates a more uniform buffer load distribution.

In Fig. 9, we observe the effect of varying the source node on retention ratio. Recall that the total number of

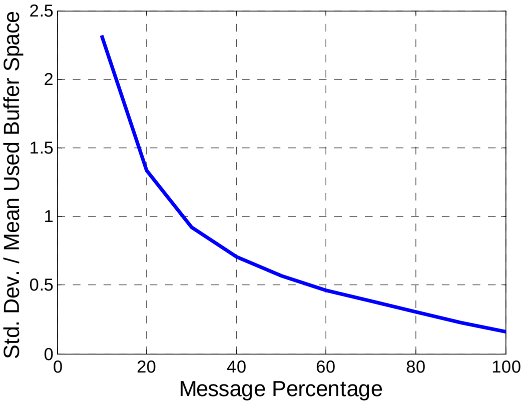

Figure 8: Std. Dev. over Mean Used Buffer Space

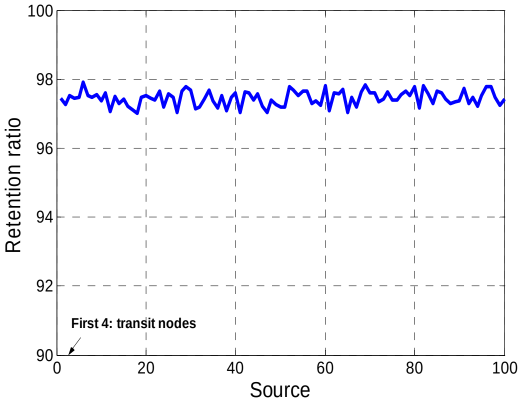

Figure 9: Effect of source change on the retention ratio

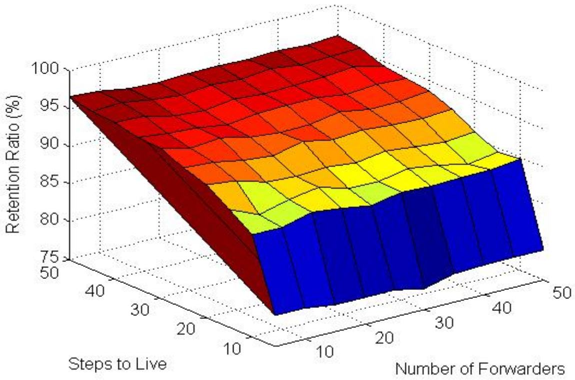

Figure 10: Effect of steps to live and number of forwarders on the retention ratio

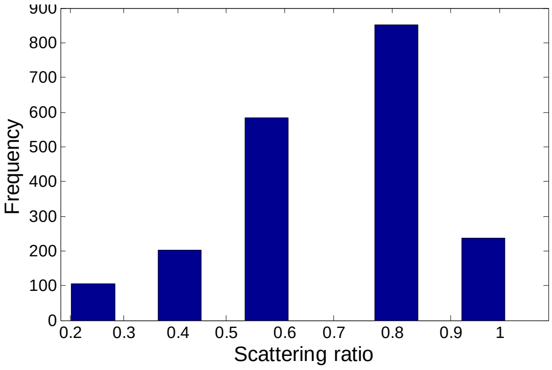

Figure 11: Scattering of bufferers to different domains

messages generated equals the total capacity of all buffers in the overlay network. Therefore, retention ratio can be at most 1 and the closer is the better. The retention ratio is quite uniform for different locations of the source which shows that our scheme is robust in this respect. What is more, the retention ratio is above 97%, that is, approximately only 3% of the messages is discarded due to buffer overflows.

The effect of steps to live and the number of forwarders parameters on retention ratio is examined in Fig. 10. The maximum value is obtained when STL is 40 and the number of forwarders is 35. Also we can infer that an increase in steps to live value has a positive effect on the retention ratio and the number of forwarders affects positively after the value of 30.

The number of bufferers *b* of a message has been set to 1 in the results given above. If *b* is set to a value greater than 1, we use the metric scattering ratio to evaluate the performance. In this case, a key aim of Stepwise Probabilistic Buffering is to minimize the average number of hops from all peers to the nearest bufferer for each message. In the simulation, we generate 2000 messages from a single source and *b* is set to 5. So, after the generation of all messages, 10,000 messages pass through the long-term buffers of the members. In the best case, the scattering ratio is 1 when 5 copies of the message are buffered in 5 different domains. We conclude from Fig. 11 that for more than half of all 2000 messages, the scattering ratio is 0.8 or 1. Namely, the 5 bufferers are selected from 4 or 5 different domains which should help in the data dissemination phase.

B. *Data Dissemination and Comparative Results*

Comparison of Stepwise Probabilistic Buffering with the hash-based approach [3] and our preliminary work Random Buffering [17] in terms of distribution of the buffering load among peers is performed. In this set of simulations, the number of peers is set to 1000, long-term buffer size of each peer is equal to 50, and 50,000 messages are generated which is equal to the system-wide long-term buffer capacity. The number of messages buffered by each peer is depicted in Fig. 12. In the hash-based and random buffering approaches, all peers have the full membership information of all the other peers so uniform distribution of buffering load is expected. In Stepwise Probabilistic Buffering, although every member has a partial view, buffering load is distributed uniformly.

We evaluate the behavior of Stepwise Probabilistic Buffering in terms of message dissemination metrics as a function of parameters of the model. We use the following performance metrics for this purpose:

- *Reliability* is the ratio of total number of received messages by peers over the total generated messages. Namely it shows how reliable the generated messages are delivered by the receivers.

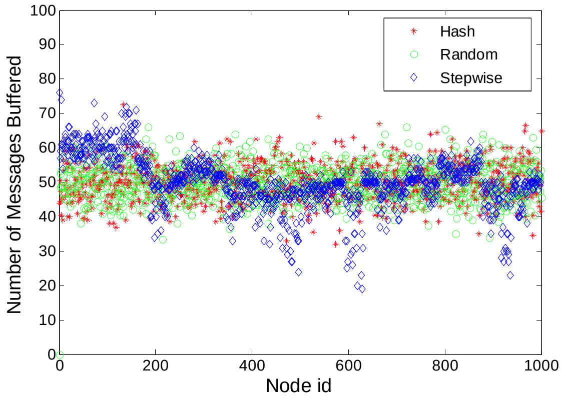

Figure 12: Number of messages buffered by each peer

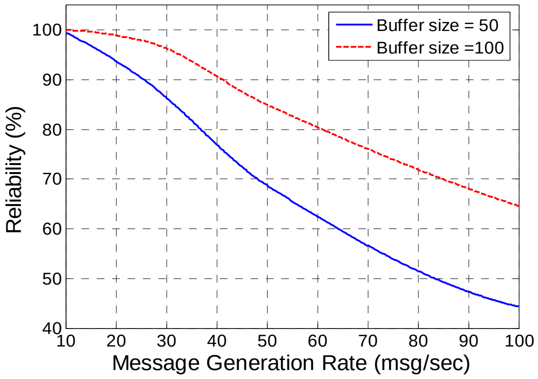

Figure 13: Message Generation Rate - Reliability

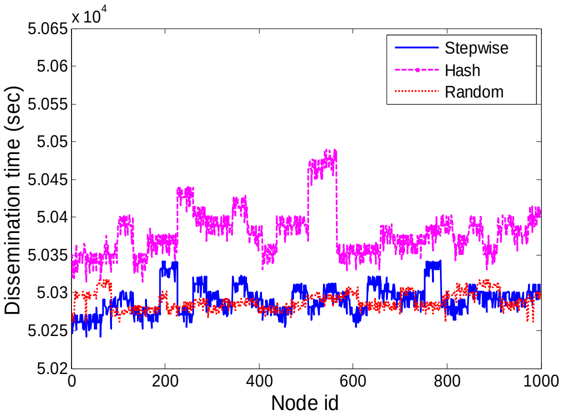

Figure 14: Content dissemination times

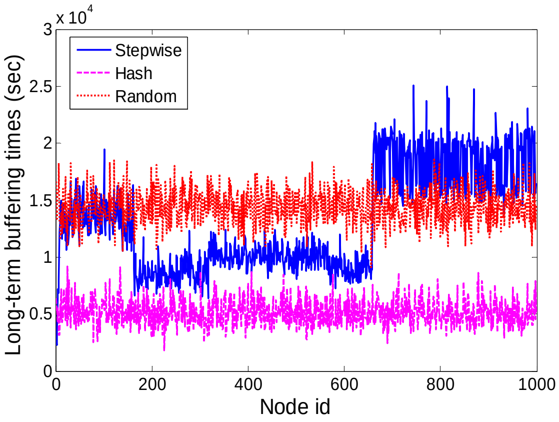

Figure 15: Long-term buffering times

- *Long-term / Short-term Buffering Time* is the mean time that a message is stored on a peer's long-term / short-term buffer.
- *Dissemination Time* is the time that passes for dissemination of the content to all peers.

First, we examine the effect of message dissemination rate on the reliability of the message dissemination. In these set of simulations, the number of peers is set to 1000. For each peer, short-term buffer size is 20 messages and long-term buffer size is 50 messages. Gossip interval is set to 100 msec. 10,000 messages are generated from a single source.

As the message generation rate is increased, the buffers of the peers are loaded and unloaded faster. Since the gossip interval stays constant for each message generation rate, the number of messages entered to the system in each gossip round grows up. Digest message size is constant for each generation rate as well. Thus, if the generation rate is raised to a certain value, state information that passes through the digest messages is updated too fast. Because of these facts, peers may not get timely information on the bufferers of some messages that they lack, or some messages may be removed from the buffers of the bufferers before they are received by some receivers. Therefore, if the message generation rate is increased keeping the other parameters constant, the reliability of dissemination is reduced. Fig. 13 shows the results of our simulations which support these facts.

When the gossip interval is increased, we observe similar effects. As we have discussed above, when the generation rate is increased the peers may not retrieve some missed messages. When the gossip interval is reduced, peers begin to inform each other about their message history more frequently. In larger gossip intervals, peers begin to discard the messages from their buffers more rapidly. Besides, probability that the bufferers remove the same messages also increases. Therefore, the reliability of the system decreases if the gossip interval is increased.

We also compare our model with the hash-based approach [3] and Random Buffering [17] in terms of dissemination time in a 1000 node scenario. In these simulations, 500,000 messages are generated from a single source. Message generation rate is 10 msgs /sec. and gossip interval is 1 sec. As shown in Fig. 14, when Stepwise Probabilistic Buffering is used, lower dissemination times than the hash-based approach are achieved. In Stepwise and Random, the bufferers are determined when a message is generated and the message is directly sent to the bufferers. However, in the hash-based approach, a peer decides to be a bufferer for a message when it receives the message through gossiping eventually. The smallest dissemination time occurs with Random which serves as a baseline for comparison. The bufferers are selected at random immediately in this approach because the sender is assumed to have a full knowledge of the overlay network.

Comparison of the mean long-term buffering time of each peer is given in Fig. 15. These results indicate that in Stepwise, a peer serves for a message for a longer time close to the average time that Random achieves. Therefore, during dissemination the availability of a message is more likely in Stepwise than the hash-based approach.

V. Conclusion

We have proposed a novel buffer management scheme Stepwise Probabilistic Buffering that distributes the load of buffering to the entire system where all peers have partial knowledge of the overlay. Our approach reduces the memory usage; it is applicable to dissemination of data to a large group of peers where epidemic dissemination idea is used. We have shown that Stepwise Probabilistic Buffering scheme distributes the buffering load uniformly to all peers, reduces the dissemination time and the buffer space of all data, improves the utilization of buffers and the reliability of dissemination. As further study, we plan to work on an analytical model to determine the optimal probability distribution for forwarding the buffering requests. In addition, multiple-source information dissemination scenarios, the behavior of our model in the case of failures as well as an adaptive buffer size mechanism that handles multiple sources and link failures will be investigated.

In this study, our assumption was the existence of an overlay among peers reflecting the properties of the underlying network topology, and we considered a transit-stub model as a good approximation of the Internet topology. As future work, we plan to integrate a module for topologically-aware overlay construction among the peers.

As an ongoing work, we are currently investigating another mechanism for bufferer selection in order to improve uniformity of buffer usage among peers. When a buffering request is received by a peer, it checks the current number of messages buffered by its neighbors. Then, it sends the request to the peer with the minimum number of messages. If the forwarding peer has the minimum value then it accepts the request and becomes the bufferer of the message. In this approach, there is no last forwarders mechanism and forwarding probabilities. Our initial results show that it leads to more uniform buffer usage among peers.

Acknowledgement. The authors would like to thank Ali Alagöz for fundamental discussions on the approach of this paper.

References

[1] Birman, K., Hayden, M., Ozkasap, O., Xiao, Z., Budiu, M., and Minsky, Y.,"Bimodal Multicast" ACM Trans. Comput. Syst. 17, May 1999, 41–88.

[2] Eugster, P. T., Guerraoui, R., Kermarrec, A.-M., and Massoulie, L. "From Epidemics to Distributed Computing" IEEE Computer Society, 2003

[3] 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.

[4] 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.

[5] 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.

[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] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In Proc. of the 2001 ACM SIGCOMM Conference, San Diego, CA, USA, August 2001.

[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In Proc. of the 2001 ACM SIGCOMM Conference, San Diego, CA, USA, August 2001.

[9] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," Comput. Sci. Div., Univ. California, Berkeley, Tech. Rep. UCB/CSD-01-1141, 2001.

[10] 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 K.

[11] 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.

[12] .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.

[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

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

[16] http://www-static.cc.gatech.edu/fac/Ellen.Zegura/graphs.html

[17] 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