Design and Analysis of a Novel Buffer Management Model for Reliable Content Dissemination

by

Emrah Ahi

A Thesis Submitted to the

Graduate School of Engineering

in Partial Fulfillment of the Requirements for

the Degree of

Master of Science

in

Computational Sciences and Engineering

Koç University

May 2007

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Emrah Ahi

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

Committee Members:

Associate Prof. Mine Çağlar, (Advisor)

Assistant Prof. Öznur Özkasap, (Co-Advisor)

Assistant Prof. Emine Şule Yazıcı

Assistant Prof. Lerzan Örmeci

Assistant Prof. Ahmet Akkaş

Date: 28/05/2007

To my parents Zerrin and Kazım and my brother Eren, I gratefully dedicate this thesis.
Thank you for always supporting me.

ABSTRACT

For supporting reliability in distributed content dissemination services, message loss recovery mechanism achieved via efficient buffer management is an indispensable component. The available approaches for buffer management concentrate on several aspects of the problem such as flow control, reducing the memory usage, providing message stability and the replacement of buffer items.

In this thesis study, we consider buffer management problem in support of large-scale bio-inspired peer-to-peer data dissemination services. Bio-inspired epidemic protocols have considerable benefits as they are robust against network failures, scalable and provide probabilistic reliability guarantees. Coupled with an efficient buffering mechanism, system wide buffer usage can be optimized while providing reliability and scalability in such protocols. We propose a novel algorithm, Stepwise Fair-share Buffering, that is shown to provide uniform load distribution in comparison to earlier approaches and reduces the overall buffer usage where every peer has the partial view of the system. A major aim of our approach is to be able to choose bufferers uniformly throughout the system so that the load of buffering will be well balanced among participating peers and the efficiency of content dissemination will be improved as a result. This also reduces the memory 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. It adjusts the buffer size to achieve message stability with a high probability.

Performance evaluation of the buffering model and extensive comparisons with earlier approaches are performed. The evaluations include scalability, reliability, adaptivity to failures and uniformity analysis. We also derive analytical results for reliability of dissemination as a function of buffer levels. These results are based on a Markov chain analysis and are evaluated numerically. Comparison with simulations shows that they provide a good lower bound for reliability. For high level of reliability values, the bounds are very close to the simulation results.

ÖZETÇE

Dağıtık içerik dağıtım servislerinde güvenilirliğin sağlanması için, etkin bir ara bellek yönetimi yoluyla başarılmış bir kayıp mesaj kurtarım mekanizması vazgeçilmez bir bileşendir. Ara bellek yönetimi konusundaki mevcut yaklaşımlar, akış kontrolü, bellek kullanımının azaltılması, mesaj dengesinin sağlanması ve bellek parçalarının yer değiştirmesi gibi çok sayıda problem bileşeni üzerinde yoğunlaşmaktadır.

Bu tez çalışmasında, geniş ölçekli biyolojiden esinlenen eşler arası veri dağıtım servislerine yönelik ara bellek yönetimi problemi ele alınmıştır. Biyolojiden esinlenen epidemik protokoller; ağ hatalarına karşı dayanıklı ve ölçeklenir olmaları ve olasılıksal güvenilirlik garantisi sağlamaları açısından kayda değer avantajlara sahiptir. Bu tip protokollerde güvenilirlik ve ölçeklenirlik sağlamasının yanı sıra, etkin bir ara bellek mekanizması ile birleştirildiğinde, sistem genelindeki bellek kullanımı da iyileştirilebilir. Önceki yaklaşımlarla karşılaştırıldığında tekdüze bir yük dağılımı sağladığı kanıtlanan ve eşlerin her birinin sistemin kısmi bir görünümüne sahip olduğu bir koşulda genel ara bellek kullanımını azaltan ve Adımsal Eşit Dağılımlı Ara Bellek olarak adlandırılan yeni bir algoritma önermekteyiz. Bu yaklaşımın başlıca hedeflerinden biri; sistem içerisindeki ara bellek tutucularının ara bellek yükü mevcut eşler arasında dengelenecek şekilde seçilmesi ve bunun sonucunda içerik dağıtımının etkinliğinin arttırılmasıdır. Bu yaklaşım; her bir mesajın ara bellek tutucusu olarak yalnızca eşlerin küçük bir alt kümesi seçildiğinden, bellek kullanımını da azaltmaktadır. Aynı zamanda, geniş ölçekli senaryolara uygulanabilir, güvenilir bir dağıtım sağlar ve dinamik sistem giriş ve çıkışlarına adapte olabilir. Ara bellek boyutunu ayarlayarak, yüksek olasılıkla mesaj dengesini sağlar.

Ara bellek modelinin başarım değerlendirmesi ve önceki yaklaşımlarla kapsamlı bir karşılaştırması gerçekleştirilmiştir. Değerlendirmeler; ölçeklenirlik, güvenilirlik, hatalara uyumluluk ve tekdüzelik analizlerini içermektedir. Ara bellek düzeylerinin bir fonksiyonu olarak dağıtım güvenilirliğine ilişkin analitik sonuçlar da türetilmiştir. İlgili sonuçlarda Markov zincir analizi temel alınmıştır ve bu sonuçlar sayısal olarak değerlendirilmiştir. Benzetimlerle gerçekleştirilen karşılaştırmalar, sonuçların güvenilirlik açısından iyi bir alt sınır oluşturduğunu göstermektedir. Yüksek düzeyli güvenilirlik değerleri için, elde edilen analitik sınırların benzetim sonuçları ile tutarlılığı gösterilmiştir.

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my thesis supervisors Associate Prof. Mine Çağlar and Assistant Prof. Öznur Özkasap for their support, valuable criticism, and endless patience throughout my study. I would like to thank Assistant Prof. Emine Şule Yazıcı, Assistant Prof. Lerzan Örmeci and Assistant Prof. Ahmet Akkaş for their valuable comments and time in reviewing this thesis as my committee members. I would like to thank to my close friend Ali Alagöz for his encouragement and support. I was fortunate to have such great friends close to me during my graduate education; Aydın Varol, Uğur Kaplan, Mert Sedef, Emre Dikmen, Murat Ötkür, and Cenk Dinç.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

ACK	Acknowledgement
BF	Buffer Fullness Ratio of a Peer
BL	Buffering Request List
CAN	Controller Area Network
CPU	Central Processing Unit
FIFO	First in First out Queue
LRU	Least Recently Used
NAK	Negative Acknowledgement
NP	Neighbor History Information
ODE	Ordinary Differential Equation
P2P	Peer-to-Peer
RMTP	Reliable Message Transport Protocol
RRMP	Randomized Reliable Multicast Protocol
SRM	Scalable Reliable Multicast Protocol
STL	Steps-to-Live

Chapter 1

INTRODUCTION

For supporting reliability in distributed content dissemination services, message loss recovery mechanism achieved via efficient buffer management is an indispensable component. In this thesis study, we consider buffer management problem in support of large-scale bio-inspired peer-to-peer data dissemination services. Coupled with an efficient buffering mechanism, system wide buffer usage can be optimized while providing reliability and scalability in such protocols. Bio-inspired epidemic protocols have considerable benefits as they are robust against network failures, scalable and provide probabilistic reliability guarantees. Hence, several distributed services such as failure detection [1], data aggregation, resource discovery and monitoring [2], and database replication [3] utilize epidemic algorithms.

Peer-to-peer (P2P) distributed dissemination applications need dissemination of content which originates from a source to a large number of peers. In an epidemic algorithm, every peer of the system is potentially involved in the dissemination of messages. Therefore, the network load is distributed to all members. Every peer ordinarily buffers each message, the information unit; it receives up to the capacity of its buffer, which is called short-term buffer in this thesis. The reliability of information delivery depends both on these values as well as on the number of participants n in the system. 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 the message. A popular distribution model based on the theory of epidemics is the anti-entropy. In the anti-entropy process, non-faulty peers are always either susceptible or infectious. Each peer periodically picks f (fan-out parameter) other peers at random, and sends them a digest including its recent message history, in other words gossips. If a randomly selected node finds out missing messages in its own history, then it requests them from the infectious nodes. The gossiping mechanism provides high resilience to problems like network failures, slow links or a failure on a single node. Eventually the message will be received by all members with high probability in O(log(n)) rounds. No mechanism is needed to detect and reconfigure from failures, unlike reactive algorithms where processes react to failures by retransmitting missing information. In anti-entropy, there is a probabilistic guarantee of delivery which 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.

While implementing protocols using epidemic algorithms, two significant issues emerge, namely, membership information and buffer management. In large-scale and dynamic group applications, it is impractical for processes to have full membership information about all other processes in the system. Hence, the group members typically have only partial views. To ensure reliability, the peers exchange messages they have buffered in their short-term buffer. As buffer capacities are limited, efficient buffer management is important in providing reliable information dissemination. In this thesis, we develop, implement and analyze novel buffering approaches for bio-inspired epidemic dissemination where each peer has only partial membership information. Depending on the rate of new information production in the system, the short-term buffer capacity of peers may be insufficient to ensure that every message is buffered long enough so that it can be forwarded an adequate number of times to achieve acceptable reliability. Setting short-term buffer capacities as large as needed for reliable dissemination is an inefficient use of network resources. On the other hand, our buffering mechanism is based on an optimization where each message is buffered for sufficiently long periods of time by only a fraction of all members to achieve high reliability. These members, called the bufferers of a message, store the message at their long-term buffer. Upon receiving a digest, a peer requests the messages that it lacks from the sender of the digest message if the short-term buffer of the sender contains them, otherwise it can request the messages 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 if any.

The available approaches for buffer management concentrate on four complementary aspects, namely reducing the memory usage [4, 5, 6, 7], flow control [8, 9, 10], providing message stability [11, 12, 13] and the replacement of buffer items [14, 15]. In approaches which optimize the memory usage, not all peers store every received message in their buffers but only some predetermined ones do. Our scheme falls into this category. In network flow control, the idea is to influence the application by regulating its rate when processes do not have enough resources and hence provide enough time to buffer and forward messages a sufficient number of times. Instead of increasing the resources, the rate of information flow is decreased. Another stream of approaches is those which detect message stability in the system. The members inform the other peers in their view about the messages they buffer. If all members detect that they have received a certain message, they all drop it from their buffers concluding that the message is stable in the system. Different policies for replacement of buffer items have also been compared in several studies. These approaches include the mechanisms for dropping which messages or when and reducing the number of nodes that buffer the messages. For deciding which message to be dropped, priorities may be given to messages. According to the priorities, the messages with low priorities are dropped when the buffer capacity is reached. These priorities can be based on age, application semantics or can be random. The age of a message is the number of times the message has been transmitted. This notion is not local to a process but to a message: the age of a message is incremented whenever the message is transmitted to a new member, and the message is tagged with its age. If the buffer of a member is full and it has to drop a message, instead of dropping a message in an arbitrary way, the member chooses the message with the highest age. The model in this study follows first in first out (FIFO) scheme, where in case of a buffer overflow the message which came first, that is the oldest message in the buffer, is dropped. Application semantics depends on the obsolescence relation which is defined by the programmer. For example message m1 makes m2 obsolete when received. Thus, the second one is dropped when the buffer size is full. An alternative way is dropping messages randomly. Also a timer can be used to drop a message. Least recently used (LRU) method is comparable to FIFO scheme [16], which has not shown significant difference in our simulations. We describe the related work in Chapter 2.

Our first model for the buffering problem, namely Stepwise Probabilistic Buffering [17] aims to distribute the load of buffering to the entire system. It provides a fairly uniform buffer distribution in a partial view scenario. However, we show that the uniformity is observed only when the number of generated messages approaches the total long-term buffer capacity of the system. If a snapshot of the system is taken when the number of generated messages is equal to a small fraction of the total long-term buffer capacity, a large deviation is observed on the buffering load of the peers. We describe Stepwise Probabilistic Buffering and analyze it using our simulation model in Chapters 3 and 4, respectively. The optimizations of the scheme help to balance the buffering load further among the peers, but they lead to overhead on the buffering request message.

Our main contribution in this thesis is a more robust scheme named Stepwise Fair-share Buffering [18] which overcomes certain difficulties associated with the probabilistic algorithm. It is a novel approach which provides truly uniform load distribution and reduces the overall buffer usage while each peer has only a partial view of the system. The load of buffering is well balanced among participating peers and content dissemination takes place efficiently. Furthermore, it is applicable to large-scale scenarios, provides reliable delivery and is adaptable to dynamic join and leaves to the system. We explain the principles of Stepwise Fair-share Buffering in Chapter 5. The buffer size can be adjusted to achieve message stability with a high probability. A discrete event based simulation model for performance evaluation of the scheme is developed. The uniformity, scalability, reliability and adaptivity of the scheme are investigated using wide range of simulation scenarios. Stepwise Fair-share Buffering is compared with the earlier approaches. These results are given in Chapter 6. Later in Chapter 7, we derive an analytical model for computing reliability of dissemination as a function of buffer levels as well as the number of bufferers. These results are based on a Markov chain analysis and are evaluated numerically. Comparison with simulations shows that they provide a good lower bound for reliability. For high level of reliability values, the bounds are very close to the simulation results. Finally, the conclusions and future work are given in Chapter 8.

Chapter 2

RELATED WORK

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. The classification of buffer management approaches is given in Fig. 2.1. In this section, we review the related work and compare with our approach.

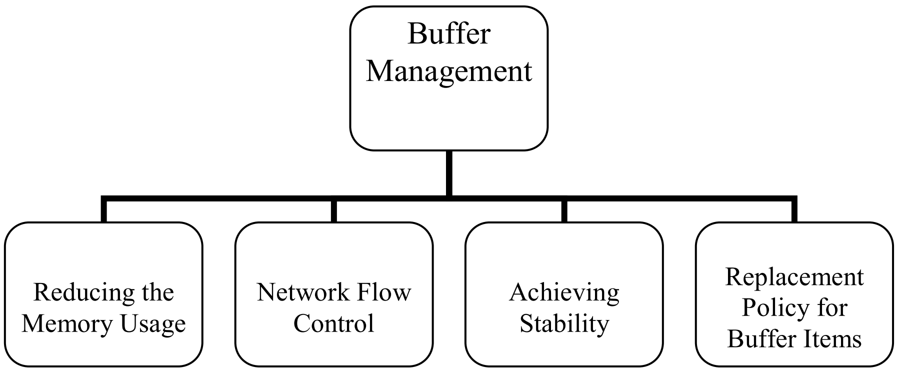

Buffer Management
Reducing the Memory Usage
Network Flow Control
Achieving Stability
Replacement Policy for Buffer Items

Figure 2.1 Classification of buffer management approaches

2.1 Reducing the Memory Usage

The pioneering study [4] 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. A commonly used identifier is [*source address*, *sequence number*]. This hash function will be described in detail in section 4.1. 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 Stepwise Fair-share Buffering, 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 novel protocol that reduces buffer requirements is Randomized Reliable Multicast Protocol (RRMP) [5] which uses epidemic error recovery. The protocol is an improvement over Bimodal Multicast [17]. In Bimodal Multicast, a receiver buffers messages for a fixed amount of time after their initial reception and then garbage collects them. In contrast, in RRMP the buffer space of the system is divided into two parts: a feedback based short-term buffer and a randomized long-term buffer. So the buffer space is reduced when compared to Bimodal Multicast. The members are grouped into local regions and the regions are formed according to their distance from the sender. A receiver has the information of the members in its local region and in its parent region. When a receiver receives a message, it keeps the message in its short-term buffer until no request arrives for this message for a certain period of time. Then, the member makes a random choice, with a predetermined probability p, to be a long-term bufferer for the message. This probability p is chosen so that the expected number of bufferers in a region is a constant C. The message is kept in the long-term buffer for a fixed amount of time if the member becomes a long-term bufferer for the message. There is a probability of discarding messages which can be requested afterwards. In RRMP, if a member detects that it has missed a message, then it sends request messages to all members in its local region. The drawback here is that search time to find the repair node can be considerably high when the number of members in the system increases. Also a message is kept in the long-term buffer for a fixed amount of time. In Stepwise Fair-Share Buffering, the messages remain in the buffers until the capacity of a buffer is reached.

A tree based reliable multicast protocol in this category is the reliable multicast transport protocol (RMTP) [6]. The protocol is designed for reliable delivery of data from one sender to a group of receivers. In RMTP a hierarchical tree-based approach is used. Receivers are grouped into local regions or domains and in each region there is a special receiver called designated receiver. Each designated receiver has the knowledge of the members in its local region and the sender. A designated receiver in each local region is responsible for sending acknowledgments periodically to the sender, for processing acknowledgment from receivers in its domain, and for retransmitting lost packets to the corresponding receivers. The sender multicasts data to all receivers but only designated receivers inform the sender about their status. Each receiver periodically sends an ACK to its designated receiver instead of sending an ACK for every received packet. This ACK contains the maximum packet number that each receiver has successfully received. However, error recovery is delayed by this periodic feedback policy. Hence, RMTP is not suitable for applications that transmit time-sensitive data. In addition, in RMTP the whole multicast session data is in the secondary storage of the repair node for retransmission. Therefore, it is not applicable to large groups or long-lived sessions.

In Scalable Reliable Multicast (SRM) protocol [7] packets are not buffered at the transport level. But, the application regenerates packets if necessary. This depends on the principle of Application Level Framing (ALF) [10] which states that networking mechanisms should be coordinated with application-level objectives. In SRM, when a receiver detects missing data, it waits for a random time determined by its distance from the original source of the data before it sends a repair request. Repair requests are multicast to the whole group just as regular data packets are. Thus, although a number of hosts may all miss the same packet, a host close to the point of failure is likely to time out first and multicast the request. Other hosts that are also missing the same packet hear that request and suppress their own request. This prevents a request implosion. Any host that has a copy of the requested data can answer a request. However, it will set a repair timer to a random value depending on its distance from the sender of the request message and multicast the repair when the timer goes off. Other hosts that had the packet and scheduled repairs will cancel their repair timers when they hear the multicast from the first host. Also in SRM, if a single link to one member of the group has a high error rate, then all members of the multicast group will contend with a multicast request and one or more multicast responses.

Another buffer management scheme which reduces memory usage is [13] where the members are organized as regions. In every region, the nodes with the most reliable links are responsible for buffering the data.

2.2 Network Flow Control

Flow control is an adaptive mechanism that deals with varying resources such as CPU speed and bandwidth in the end hosts. Buffer optimization techniques that fall into this category adjust the rate on the network so that the buffer overflows at the end hosts are minimized. In an earlier study of Mishra and Wu [8] which is a survey paper, they investigated the effect of buffering rate and flow control in some ACK based and NAK based reliable multicast protocols. It is concluded that rate-based protocols are the best since they are more scalable and have better reliability.

In the NAK based retransmission control scheme given in [9], the sender reduces its transmission rate whenever it receives too many NAKs from the receivers. The sender also keeps a log of its past transmission rates to prevent high decrease in the rate. So, this mechanism helps to minimize the buffer overflows at the receivers.

A different idea explored in [10] 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 that 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 of information production is adjusted according to the process with the smallest buffer space.

2.3 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 [12], there is a stability detection algorithm for discarding safe messages from the buffers. The members are partitioned into groups and every node is included in the error recovery. 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 [13] 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 [11] 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.

2.4 Replacement Policy on Buffer Items

Network Friendly Epidemic Multicast [14] 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 [15] 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.

2.5 Structured Peer-to-Peer Networks

The approaches explained up to this point have been built on unstructured networks. They are not embedded with a logically deterministic structure for organizing and managing the peers. These systems employ a message flooding for searching interested items. To prevent the high cost of flooding they use a time to live mechanism for the messages. On the other hand structured P2P protocols such as Chord [21], CAN [22], Pastry [23], Tapestry [24], manage the peers with an implicit and deterministic structure. These protocols offer a management on participating peers and published data items. CAN employs a multidimensional coordinate space, Chord is based on a ring, in Pastry and Tapestry hypercube is used. These systems name the participating peers and available data items with a distributed hash function. The data items are identified by hashing keys. A data item with hashing key k is managed by the peer whose hashing key is closest to k. To retrieve a data item with hashing key k , the request is forwarded to intermediate nodes whose hashing key are closer and closer to k. If a uniform hashing function is used, the number of stored data items will be approximately equal at each peer. By this way, the buffer load on the peers can be balanced.

Chord [21] embeds peers with a single hash address space. As explained above, the data items and the IP-address of peers are hashed with a specified hash function. The hash space is organized as a circular structure. All the participating peers are arranged in ascending order in a circle. Chord [21] assigns keys to nodes with *consistent hashing*. With 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. In Chord, there is a concept of successor function. For key k, successor (k) corresponds to the first actual node following k around the circle. Linear searching in a large scale network is inefficient so in Chord every peer uses a 'finger table'. Finger table has m entries, indexed by 0 through m-1 each one pointing to an actual node. Each node stores the IP addresses of relatively small number of nodes. In Fig. 2.2, the idea of Chord is illustrated with a simple scenario. In the illustration 1, 3, 6, 10 and 13 are actual nodes. The tables attached to the nodes represent the finger tables of the corresponding nodes. The first column represents the keys stored in the node and the second column represents the IP address of the key. For example, to look at key 15 from node 3, the finger table is consulted. The closest predecessor of 15 is 11, so the request is forwarded to IP address of 11's entry namely that of node 13. Node 13 sees that node 15 lies between its IP address and its successor 1. So it returns the IP address of node 1.

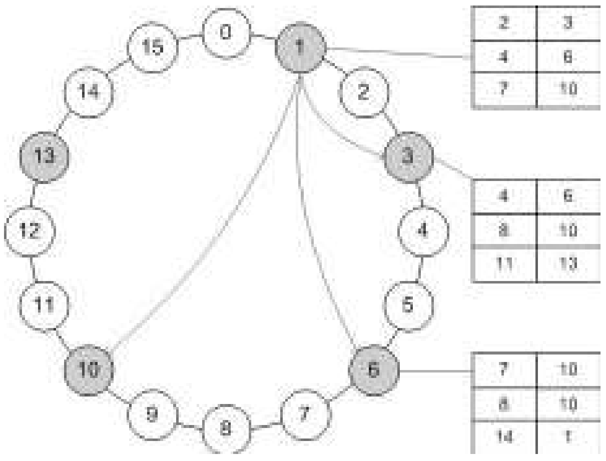


Figure 2.2. Illustration of Chord Protocol

2.6 Survey on Analytical Studies

In this section, we review prior analytical studies in the context of buffer management. These are classified according to the approach they adopt, namely management of history buffer size, determination of buffer hit rate and ordinary differential equation models.

2.6.1 Management of History Buffer Size

In the study of B. Koldehofe [25], the focus is on the size of history buffer of every peer when a single source epidemic dissemination paradigm is used. It is highlighted that the size of history buffer must be chosen large enough to guarantee safe delivery and not to give rise to multiple deliveries of the same message to the application. The buffering mechanism is designed as a queuing system in which new coming messages are added to the queue as a random process. Let *m* be the number of rounds an event stays at most in the system and *n* be the number of nodes. Let $[t_a, t_s]$ be the time interval of length *m* and $X_{i,j}$ be the random variable representing process *j* admits a new gossiping event at time $t_a + i$. It is assumed that all $X_{i,j}$ occur independently, $P\{X_{i,j} = 1\} = p$ and $P\{X_{i,j} = 0\} = 1 - p$. Thus, the total number of gossiping events in $[t_a, t_s]$ is $X := \sum_{j=1}^{n} \sum_{i=1}^{m} X_{i,j}$. The process that describes the new incoming event is binomially distributed and the expected number of events in the queue in the interval $[t_a, t_s]$ is $E[X] = pnm$. According to the analysis, the history buffer size must be chosen greater then *pnm* to guarantee safe delivery. As a second step, a bound for the buffer size is determined to minimize the multiple deliveries of the same event. This bound is computed using the Chernoff bound for binomial distribution [26]. If the buffer size is chosen greater than *2pnm*, multiple deliveries of the same event to the application are minimized.

Also, in the study [25] reliability properties of FIFO buffering scheme, estimated time to terminate approach and estimated time to potential approaches are compared. It is concluded that the scheme that uses the "estimated time to terminate" approach for buffering shows the best performance among others. This approach is based on the estimation of rounds a message needs to reach all participants and counting the number of hops a message has performed. The estimated potential approach is based on the fact that for a constant $c > 1$, placing $cn\log(n)$ balls uniformly at random into n bins is sufficient for every bin to receive at least one ball with high probability.

2.6.2 Determination of Buffer Hit Rate

In [15], performance of Least Recently Used (LRU) buffering policy is evaluated and a model is developed for determining buffer hit rate on mobile devices for epidemic information dissemination. Buffer hit rate refers to the rate at which an item can be found in a buffer. In LRU scheme, a new coming message is placed on the first position and the message at the rear is discarded. 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 the model, D data items are partitioned such as $D(1), D(2), ..., D(K)$ where K is the number of distinct keys. Each key k matches a fraction of $\beta(k)$ of the data items. Let $b(k, j)$ denote the number of items of partition $D(k)$ in the top j positions of the LRU stack. Thus $\frac{b(k, j)}{\beta(k)D}$ is the hit probability in the top j stack positions. Let $r(k, j)$ denote the rate for pushing down items of partition $D(k)$ from stack position j to stack position j+1. Therefore $r(k, j) = \lambda\alpha(k)\left(1 - \frac{b(k, j)}{\beta(k)D}\right)$. Let $p(k, j)$ denote the probability that an item of partition $D(k)$ is located at position j in the LRU stack. It is computed approximately as $p(k,j) \approx \frac{r(k,j)}{\sum_{n=1}^{K} r(n,j-1)}$. So $b(k,j)$ can be determined as $\sum_{n=1}^{j} p(k,n)$. As a result the hit rate of LRU scheme is computed as

$$HR_{LRU} \approx \sum_{k=1}^{K} \frac{\alpha(k)b(k,B)}{\beta(k)D} \tag{1}$$

In the study, this idea is extended to compute the hit rate of LRU scheme for epidemic dissemination on mobile devices. Hit rate for the protocol 7DS is computed as:

$$HR_{7DS} = \sum_{k=1}^{K} \alpha(k)\Big(1-\big(p_{local}(k,B)\big)\big(1-p_{origin}(k)\big)\big(1-p_{remote}(k)\big)\Big) \tag{2}$$

where $p_{local}(k,B)$ is the probability for a hit for key *k* in the local buffer, $p_{origin}(k)$ is the probability for retrieving an item matching key *k* from the origin device and $p_{remote}(k)$ is the probability for retrieving an item matching key *k* from a remote device other than the origin device.

In [14], approximate analytical models for predicting the buffer hit rates for the LRU and FIFO schemes are developed. The study explained above uses the LRU model. For FIFO replacement policy, the buffer is considered as a queue where the item at position *B* is thought as the head of the queue and the item at the position 1 is thought as the tail. Parameter *B* is the buffer size. Similarly, *D* data items are partitioned into K partitions such as $D_1, D_2, ..., D_K$ where K is the number of distinct keys. Let $\underline{X}_n = (X_{1,n}, X_{2,n}, ..., X_{B,n})$ denote the state of the buffer after n^{th} request and $X_{i,n}$ denote the occupancy of the i^{th} entry in the buffer. Let $Y_{k,n}$ denote the number of items from partition D_k after n^{th} request and $Y_{k,n} = \sum_{i=1}^{B} 1(X_{i,n} = k)$ where $1(P) = 1$ if the predicate *P* is true. The authors are interested in $Y_k = \lim_{n\to\infty} Y_{k,n}$ which is the number of items from partition D_k in the steady state. In the FIFO policy, if a request is to an item already in the buffer, then the buffer remains unchanged. If a request is to an item not in the queue, then the item is placed in position 1 (tail) and all of the items within the buffer are removed one position to the rear. Let *R* be the probability that the item is removed from the buffer if a request is served. This probability is equal to the probability that a new item is brought in so $R = \sum_{k=1}^{K} \alpha_k \left(1 - \frac{E[Y_k]}{D_k}\right)$. Then the probability that a new item from partition *k* is brought in is $R\frac{E[Y_k]}{B}$. The probability that an item is brought in is $\alpha_k \left(1 - \frac{E[Y_k]}{D_k}\right)$. If the probabilities are equated, the expected number from partition D_k is obtained as $E[Y_k] = \frac{D_k}{1 + \frac{RD_k}{\alpha_k B}}$.

2.6.3 Ordinary Differential Equation Models

In another study [27], epidemic dissemination is modeled by an ordinary differential equation. Different variants of epidemic dissemination are studied and performance of epidemic dissemination with different buffer management schemes is modeled. Suppose *N* is the number of nodes, L is the average lifetime of a packet, $I(t)$ is the number of infected nodes at time t and λ is the packet rate of the data flow. Thus $\int_0^\infty \frac{I(t)}{L} dt$ gives the average number of copies of a packet. Average number of packets in the system is *NλL* by Little's Law. Therefore, buffer occupancy in the network is $E[Q_t] = \int_0^\infty I(t)Ndt$ and buffer occupancy per node is $E[Q] = \lambda \int_0^\infty I(t)dt$. In this model, storage capacity of the nodes is assumed as infinite. In the study, also models for different buffer management schemes such as drophead, droptail and drophead with high priority for source packet are developed. In droptail, when a peer's buffer is full, it will not accept any packets. Thus, the loss probability is equal to the probability that a peer's buffer is full. This probability P_d is estimated using probabilistic forward ODE $\frac{dI}{dt} = \beta pI(N - I)$ with p=1- P_d where p is the forwarding probability, N is number of nodes in the system and I is the number of infected nodes. Drophead policy behaves similar to the FIFO scheme. Let $S(t)$ denote the number of susceptible nodes at time t and $I_i(t)$ denote the average number of infected nodes where the copy of the packet is the i-th newest packet. The following equation is used to model the spreading for this case: $\frac{dS}{dt} = -\beta S \sum_{1 \le i \le B} I_i(t)$ where B is the buffer size per node and β is the infection rate.

In drophead with high priority for source packets, if a source packet arrives to a node with a full buffer, the node drops oldest relay packets, then the oldest source packets. If a relay packet arrives to a full buffer, the node deletes the oldest relay packets; if all packets in the buffer are source packets, the relay packets are refused. For modeling of this scheme the following equation is used: $\frac{dS}{dt} = \beta(1 - P_f)S \sum_{1 \le i \le B} (I_i^S + I_i)$ where $I_j^S(t)$ denotes the probability of source node's copy of the packet is the j-th newest source packet in the buffer and P_f is the probability that a node's buffer is filled with its own packets. According to the comparative numerical results for different buffer sizes, drop tail causes the highest and drophead with high priority for source packets causes the least drop probability among these three approaches.

2.7 Exploited Ideas in Stepwise Fair-share Buffering

This thesis proposes an efficient buffering technique Stepwise Fair-share Buffering that uniformly distributes the buffering load to the entire system where members have only a partial view of the membership. The scheme aims to reduce the system wide buffer space. An explicit flow control mechanism is not arranged for the scheme. Members hold the history of messages up to a certain value and this value is chosen large enough to provide stability. A "first in first out" policy equivalent to age-based purging is implemented in the case of a buffer overflow. LRU scheme is deployed to the scheme as well and no significant difference is found. The mechanism is applicable to large scale scenarios, provides reliable delivery and is adaptable to dynamic join and leaves to the system.

In the scheme, the messages originate from a source and disseminated by an epidemic protocol. In epidemic multicast protocols data is propagated via gossiping. In the protocol, the repair phase works with the data dissemination phase. Namely, during gossiping if a member receives a digest message, then it detects the messages and it gets the missed message from the system. In our protocol, each peer periodically selects f (fan-out) 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.

Chapter 3

STEPWISE PROBABILISTIC BUFFERING

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 technique 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 Stepwise

Probabilistic Buffering which takes place concurrently with epidemic data dissemination. The major aim is to distribute the buffering load to the entire system evenly.

In this chapter, details of the Stepwise Probabilistic Buffering method are given. We first describe the principles of Stepwise Probabilistic Buffering. After that the long-term and short-term buffering schemes are explained. Then, information about the network simulation topologies used for analyzing the buffer management scheme is given. Subsequently, the data dissemination part of the protocol described and the optimizations done to increase the uniformity of the scheme are explained. The parameters, data structures, message formats and algorithms of the scheme are given at the end of the chapter.

3.1 Principles of the Stepwise Probabilistic Buffering

Stepwise Probabilistic Buffering is designed to use the buffers of peers effectively where 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 the study's scheme 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.

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 significant part of the stated method 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. 3.1 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 then epidemic data dissemination takes place.

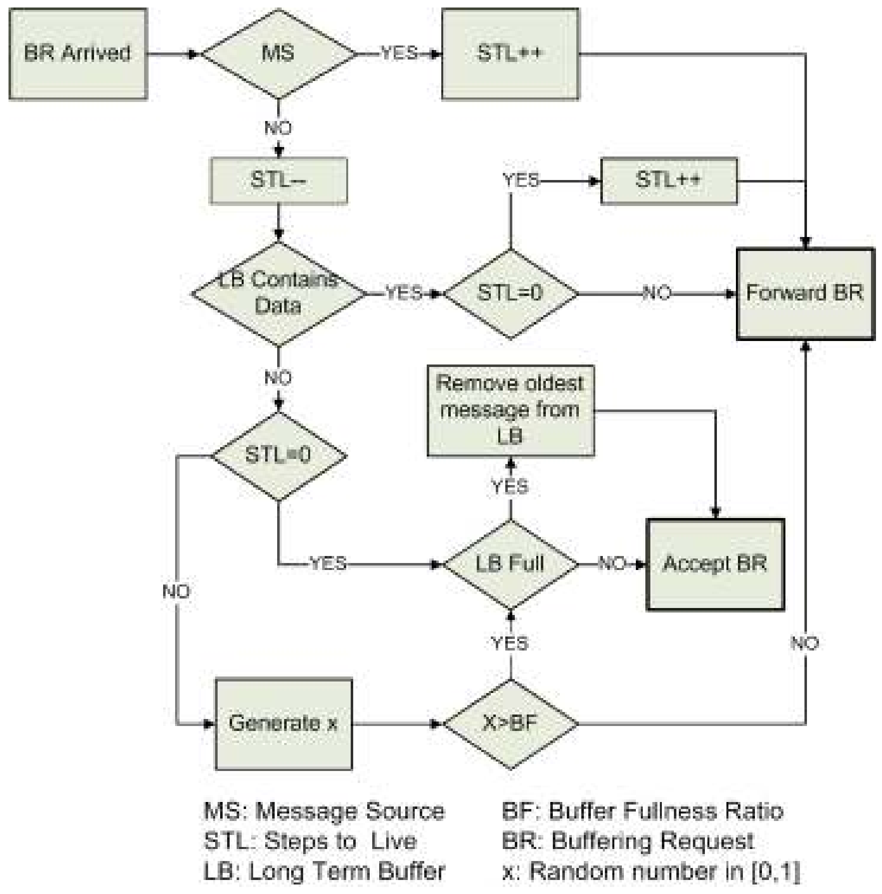


Figure 3.1 Flow chart for determining the bufferers

In Fig. 3.2 a)-d), an illustration of Stepwise Probabilistic Buffering on a simple network is given. In this example, parameter *b*, number of bufferers per message is set to 2. Assume that the partial view of the source node is composed of node 1, 2 and 3. The source node sends 2 buffering request messages to 2 randomly chosen nodes (node 1 and node 2) from its neighborhood as shown in Fig. 3.2.a). The percentage given for a node represents the buffer fullness ratio. For example 40 % and 75 % of the long-term buffers of nodes 1 and 2 are full respectively. When node 1 receives the buffering request, it generates a random number between 0 and 1, and then compares the number with its *BF* 0.4. Assume that the random number is greater than 0.4, node 1 becomes the bufferer of the message as shown in Fig. 3.2.b). Thus, it announces to the source that it has become the bufferer of the message. On the other hand, when node 2 receives the buffering request, the random number it has generated is less than its *BF* 0.75. Then, it forwards the buffering request to a randomly chosen neighbor node 4. The buffer fullness ratio of node 4 is 0.25. Node 4 becomes the second bufferer of the message. Afterwards, it announces to the source that it is a bufferer of the message as well in Fig. 3.2.c). Lastly, in Fig. 3.2.d) the source node piggybacks the bufferer ids to the message and sends them to the bufferer nodes.

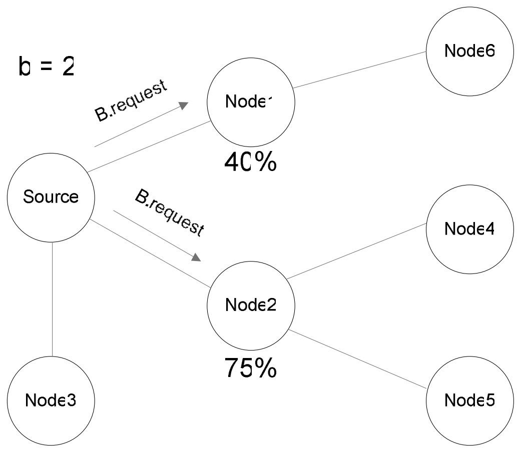


Figure 3.2 a) Sending buffering requests

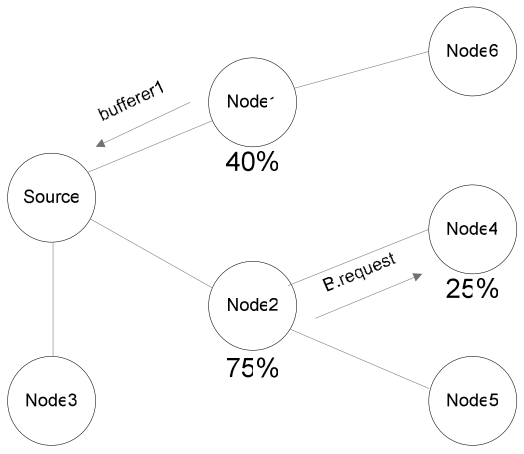


Figure 3.2 b) Forwarding buffering request

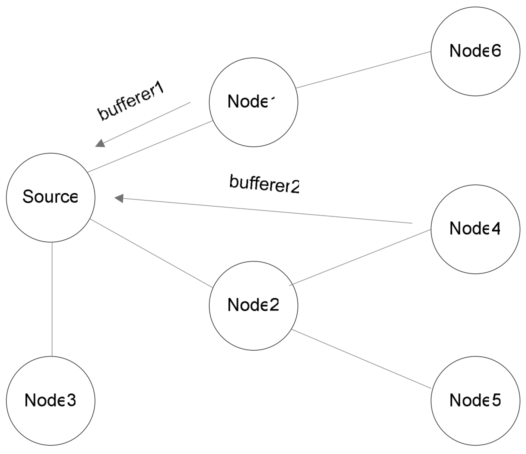


Figure 3.2 c) Bufferer announcements

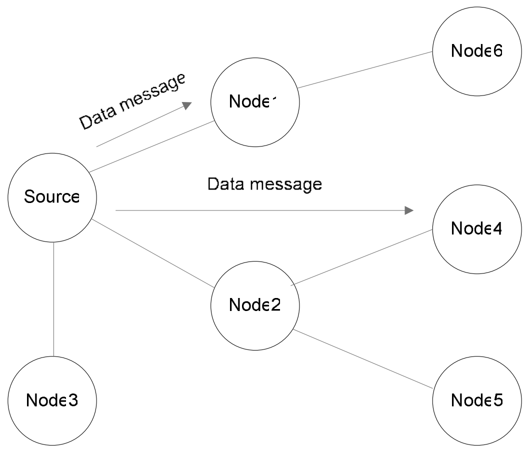


Figure 3.2 d) Sending data to bufferers

3.2 Long-term and Short-term Buffering

In Stepwise Probabilistic Buffering, there is a two phase buffering algorithm. Every peer has a short-term buffer for gossiping and long-term buffer for retransmissions. Message discarding policy is not time dependent like [6] or [17]. In [6] and [17], every message in the buffer has a predefined duration to be discarded. FIFO message discarding policy is used in both of the buffers. 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 of the capacity of the buffer is reached. Also LRU message discarding policy is deployed to compare the performance with FIFO policy. In LRU scheme, a new coming message is placed on the first position and the message at the rear is discarded. 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.

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.

When a message is generated, a set of bufferers for the message is determined by the stepwise algorithm and ids of these bufferer nodes are piggybacked to the message as explained in the previous section. When the bufferers of the messages are determined, the messages are directly forwarded to the bufferer nodes by the source. In addition, number of bufferers must be chosen large enough not to increase the overhead in the system. The bufferer processes hold the corresponding messages in their long-term buffers infinitely if there is a buffer space. The long-term buffer is used for the retransmission of missed messages. If a process detects that it has missed a message, it can request the message from one of the bufferers of that message. The parameters must be chosen so that the probability that a missed message is removed from the long-term buffers of all bufferers and there exists a process missing the message is small.

3.3 Network Topology

Existence of an overlay among peers reflecting the properties of the underlying network topology is assumed, and a transit-stub model as a good approximation of the Internet topology is considered in the scheme. 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. View of a sample transit-stub overlay is given in Fig. 3.3.

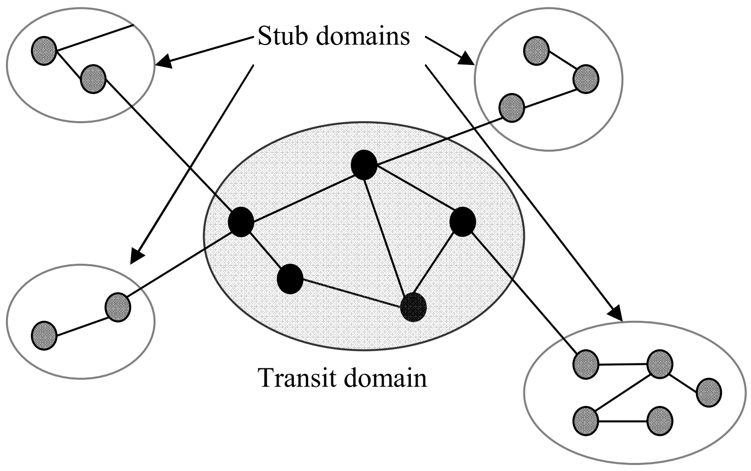


Figure 3.3 Overlay topology

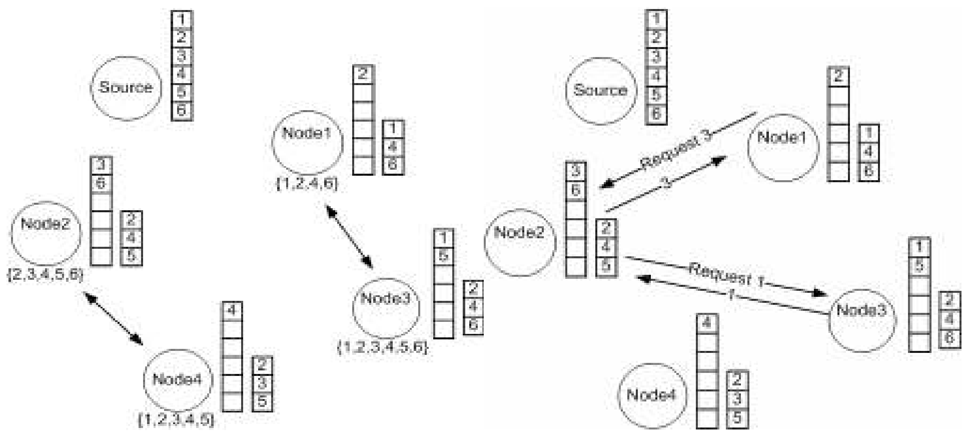
3.4 Data Dissemination

A popular distribution model based on the theory of epidemics is the anti-entropy [28]. 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, a pull-based approach is used in which data dissemination is triggered by susceptible peers when they are picked as gossip destinations by infectious peers.

The messages are disseminated to all members epidemically by the anti-entropy model. At every predefined time interval called *gossip interval*, all members choose *f* (fan-out) peers randomly and then send the information of the messages received up to that time. 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. The events, parameters and data structures are listed in the tables 3.1-3.3 at the end of the chapter. 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 3.4.a and b 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. 3.4.a, 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. 3.4.b. 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.



(a) (b)

Figure 3.4 Illustrating Stepwise Probabilistic Buffering

a) Gossiping b) Requesting missed messages

3.5 Improvements

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. On the other hand, in case of large STL, there is a disadvantage 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.

3.5.1 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. Typically, this *n* value is chosen about the size of a stub domain. 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. If a member receives a buffering request, the member writes its id to the buffering request and sends it to a random neighbor. If the last forwarders list, size of which is set as a parameter, is full, the receiving node deletes the id of the node that is at the rear of the list and writes the id of itself at the front of the list. When a member receives a buffering request, it checks the forwarders of the buffering request and chooses the destination node among its neighbors excluding the ones in the last forwarders list.

Fig. 3.5 illustrates the mechanism with an 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.

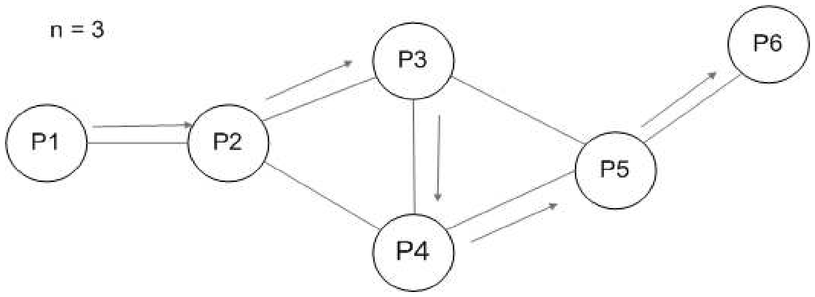


Figure 3.5 Example of last forwarders

3.5.2 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.

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. 3.6 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 *Pxy* 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. 3.6 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 a higher probability of forwarding the request is assigned to transit neighbors than forwarding the request to stub neighbors.

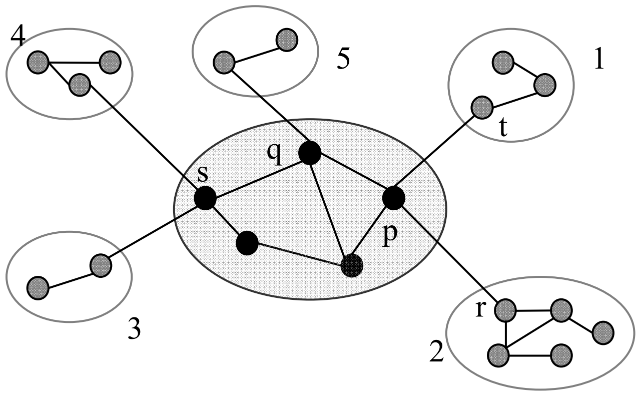


Figure 3.6. Forwarding probabilities

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. 3.6, 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.

3.6. Events, Variables, Data Structures and Message Formats

In this section we give descriptions for the events, variables and data structures of the Stepwise Probabilistic Buffering model in Tables 3.1, 3.2 and 3.3. There are four types of messages namely data, gossip, buffering request and request, used. The message formats and their descriptions are given in Table 3.4.

Table 3.1. The events of the scheme

Event	Description
Bufferer Selection	Selection of bufferer ids of a data message
Long-term Buffer Insertion	Insertion of a data message into the long-term buffer
Short-term Buffer Insertion	Insertion of a data message into the short-term buffer
Buffering Request Reception	Reception of a buffering request message
Buffering Request	Transmission of a buffering request message
Data Generation	Generation of data by the source node
Data Reception	Reception of a data message
Gossip Propagation	Gossip dissemination procedure operated periodically
Digest Message Reception	Reception of a digest message of a neighboring node
Request Message Reception	Reception of a request message

Table 3.2. The special variables

Variable	Description
Message_Id	Unique id of each data message
Gossip_Round	Gossip round counter increased in each gossip round
Fan-out	Number of nodes chosen for gossiping each gossip round
Number_Of_Bufferers	Number of bufferer nodes for the data messages
Generation_Interval	Time interval of data generation determined by the source node
Digest_Size	Number of entries in the digest message
Long-term_Buffer_Capacity	Number of messages that can be stored in the long-term buffer
Short-term_Buffer_Capacity	Number of messages that can be stored in the short-term buffer
Buffer_Fullness	Number of messages over long-term buffer capacity
Number_of_Last_Forwarders	Number of nodes that the buffering request would not be sent
Steps_To_Live	Max number of hops the a buffering request can travel
STL_Counter	Remaining lifetime of buffering request as number of hops
Source_Prob$_{tt}$	Prob. of sending from transit to transit node if node is source
Non-source_Prob$_{tt}$	Prob. of sending from transit to transit node if node is not source
Non-Source_Prob$_{ts}$	Prob. of sending from transit to stub node if node is not source
Prob$_{st}$	Prob. of sending from stub to transit node
Forwarder-Prob$_{st}$	Prob. of sending from stub to transit when every neighbor is one of the *LAST_FORWARDERS*

Table 3.3. The data structures

Data Structure	Description
DATA_MESSAGE	Data message received
DIGEST_MESSAGE	Digest message to be send
REQUEST_MESSAGE	Request message to be send for data reception
BUFFERING_REQUEST	Buffering request message sent by the source node
LONG_TERM_BUFFER	Long-term buffer of the current node
SHORT_TERM_BUFFER	Short-term buffer of the current node
MESSAGE_ENTRY	Message id and bufferers of the message
RECEIVED_MESSAGES	Message entries of received messages
ACCEPTED_REQUESTS	List of buffering requests accepted for buffering
NEIGHBOR_LIST	List of neighboring nodes in the partial view of current node
LAST_FORWARDERS_LIST	List of nodes that forwarded the buffering request

Bufferer Id: The id of one of the bufferers corresponding to the message.

Indicator value: Boolean value for representing the existence of the corresponding message

STL value: Remaining life of the buffering request as number hops

Source Id: Unique id of the source of the message

Last Forwarders: Ids of nodes that forwarded the request

Size of Message: The size of the payload

Table 3.4. Message formats

***DATA_MESSAGE*:**					
Message type *1 octet*	*2 octets*	Message Id 1 octet	Bufferer Id *1 octet*	Data Contents *1024 octets*	
***GOSSIP_MESSAGE*:**					
Message type *1 octet*		Gossip Contents [(message id, bufferer id, indicator value), (message id, bufferer id, indicator value) ...] *Max. 2048 octets*			
***BUFFERING_REQUEST_MESSAGE*:**					
	2 octets	Message Id 1 octet	STL value 1 octet	Source Id 1 octet	Last Forwarders Max. 1024 octets
***REQUEST_MESSAGE*:**					
Message type *1 octet*	*2 octets*	Request Contents *Max. 1024 octets*			

3.7. Algorithms for Determining Bufferers and Data Generation

The following are the algorithms for each event of the bufferer determination phase, described in the previous section.

Bufferer Selection:

```
Choose Fan-out destinations from NEIGHBOR_LIST
For all destinations;
    Send BUFFERING_REQUEST
End for
If (All BUFFERING_REQUESTs returned) then;
  Piggyback the bufferer ids to the DATA_MESSAGE
  Send the DATA_MESSAGE to Fan-out members in NEIGHBOR_LIST
```

Long-term Buffer Insertion:

```
If (size of the LONG_TERM_BUFFER = long-term_buffer_capacity) then;
  Remove the last DATA_MESSAGE from LONG_TERM_BUFFER
Endif
If (BUFFERING_REQUEST of the DATA_MESSAGE  is in the ACCEPTED_REQUESTS ) then;
  Add the new coming DATA_MESSAGE to LONG_TERM_BUFFER
```

Short-term Buffer Insertion:

```
If (size of the SHORT_TERM_BUFFER = short-term_buffer_capacity) then;
  Remove the last DATA_MESSAGE from SHORT_TERM_BUFFER
Endif
Add the new coming DATA_MESSAGE to SHORT_TERM_BUFFER
```

Buffering Request Reception:

```
If (Is_Message_Source (BUFFERING_REQUEST)) then;
  Increase the STL_Counter of BUFFERING_REQUEST
  Buffering Request Transmission (BUFFERING_REQUEST)
Else
  Decrease the STL_Counter of BUFFERING_REQUEST
  If (LONG-TERM_BUFFER contains DATA_MESSAGE) then;
    If (TTL_Counter=0) then;
      Increase the STL_Counter
    Else
      Buffering Request Transmission (BUFFERING_REQUEST)
    Endif
  Else
    If (TTL_Counter=0) then;
      If (size of the LONG_TERM_BUFFER = long-term_buffer_capacity) then;
        Remove the last DATA_MESSAGE from LONG_TERM_BUFFER
      Endif
       Add the BUFFERING_REQUEST to ACCEPTED_REQUESTS
    Else
      Generate a random number between 0 and 1
      If (Generated number > Buffer_Fullness) then;
        If (size of the LONG_TERM_BUFFER = long-term_buffer_capacity) then;
          Remove the last DATA_MESSAGE from LONG_TERM_BUFFER
        Endif
        Add the BUFFERING_REQUEST to ACCEPTED_REQUESTS
      Else
        Buffering Request Transmission (BUFFERING_REQUEST)
      Endif
  Endif
Endif
```

Buffering Request Transmission:

```
Find the members in NEIGHBOR_LIST different from LAST_FORWARDERS_LIST
If (Size of different members is not zero) then;
  If (Node is a transit node) then;
    If (Node is the message source) then;
       Send the BUFFERING_REQUEST to a transit neighbor with prob. $Source\_Prob_{tt}$
    Else
       Send the BUFFERING_REQUEST to a stub neighbor with a prob. $Non\text{-}Source\_Prob_{ts}$
       Send the BUFFERING_REQUEST to a transit neighbor with a prob. 1-#of stubs*$Non\text{-}Source\_Prob_{ts}$
  Else
     Send the BUFFERING_REQUEST to a transit neighbor with prob. $Prob_{st}$
  Endif
Else
  If (Node is a transit node) then;
    Send the BUFFERING_REQUEST randomly to a node from NEIGHBOR_LIST
  Else
    Send the BUFFERING_REQUEST to a transit neighbor with $Forwarder\text{-}Prob_{st}$
    Send the BUFFERING_REQUEST to a stub neighbor randomly
   Endif
Endif
```

The following are the algorithms for each event of the data generation phase.

***Data Generation*:**

```
Short-term Buffer Insertion (DATA_MESSAGE)
Choose destinations from NEIGHBOR_LIST
For all destinations;
  Create a BUFFERING_REQUEST
  Set the STL_Counter of BUFFERING_REQUEST as Steps_To_Live
  Buffering Request Forwarding (BUFFERING_REQUEST)
End for
Buffering Request Reception (BUFFERING_REQUEST)
Bufferer Selection (DATA_MESSAGE)
```

Data Reception:

```
If (DATA_MESSAGE is not duplicate ) then;
  Add MESSAGE_ENTRY to RECEIVED_MESSAGES
  If (Is_bufferer(DATA_MESSAGE) ) then;
    Long-term Buffer Insertion (DATA_MESSAGE)
  Else
    Short-term Buffer Insertion (DATA_MESSAGE)
Endif
```

Gossip Propagation:

```
If (size of RECEIVED_MESSAGES < Digest_Size) then;
  Put all MESSAGE_ENTRIES to the DIGEST_MESSAGE
Else
  Put the last Digest_Size MESSAGE_ENTRIES to the DIGEST_MESSAGE
Endif
Choose Fan-out destinations from NEIGHBOR_LIST
For all destinations;
  Send the DIGEST_MESSAGE
End for
```

Digest Message Reception:

```
Compare the DIGEST_MESSAGE with RECEIVED_MESSAGES
For all missing messages
  Send REQUEST_MESSAGE to the digest sender
End for
```

Request Message Reception:

```
If (Is_bufferer (DATA_MESSAGE) ) then;
  Scan the long-term_buffer
  Send the DATA_MESSAGE to the request sender
Else
  Scan the short-term_buffer
  Send the DATA_MESSAGE to the request sender
```

Chapter 4

SIMULATION MODEL AND ANALYSIS OF STEPWISE PROBABILISTIC BUFFERING

In this chapter, details of our simulation model and simulation results of Stepwise Probabilistic Buffering are given. First, the model that is used in the simulations is described. After that, the simulation environment and the steps of topology generation are explained. In the last section, simulation results obtained to evaluate the performance of Stepwise Probabilistic Buffering are presented.

4.1 Simulation Model

In order to analyze the performance of Stepwise Probabilistic Buffering, a discrete time event based simulation model is developed. In this model, time is incremented via discrete intervals and at the end of every interval; occurrence of new events is checked. If a new event is encountered, it is processed accordingly. This process continues until all simulation data are disseminated. The basic structure is of the form:

```
While dissemination not finished:
    t = t + dt
    Update the system state for the new time interval
    state(t) = old_state(t)+ changes
```

The events (explained in the previous chapter) are placed in a queue according to their times of occurrence.

Before the development of simulation software, we examined existing P2P network simulators. In the remainder of this section we overview some of these simulators.

Peersim [29] is a Java based search framework that allows modeling of P2P overlay search algorithms. To provide scalability and focus on self-organization properties of large scale systems, some assumptions have been made in this simulator. These assumptions include ignoring the details of the transport communication protocol stack. SimP^2 [30] is a graph-based simulator for analysis ad-hoc P2P networks. The analysis is based on uniform random graph model, and is limited to study basic properties such as reachability and nodal degree. P2Psim [31] is another simulator that provides overlay lookup, join and leave. It provides no support at the network level for controlling or modeling bandwidth. Also it does not support IP layer topology. Only end to end delays are calculated. There are only lookup and join methods for the nodes that are to be implemented.

4.2 Simulation Environment and Topology Generation

Simulations for Random Buffering [32], Stepwise Probabilistic Buffering [17], and Hash-based approach [4] are implemented with Java programming language. Java S.D.K. 1.5.0 is used and the simulations are run on machines with 2 GB RAM and 2.4 GHz CPU speed.

GT-ITM (Georgia Tech. Internetwork Topology Tool) [33] is used for transit-stub topology generation in simulations. This software package implements a collection of topology generation methods, including standard random graphs, Waxman's variant on random graphs, and the transit-stub method. The transit-stub uses the other methods to build up a topology whose high-level structure arguably reflects the high-level structure of the Internet, and it is probably the most widely used method in GT-ITM. In the simulator, a Linux script is used for generation of topology. The user enters number of transit domains, stub domains, average number of nodes in a transit or a stub domain as the program parameters. The probabilities of having a link between any two nodes in a transit domain and stub domain are set by the user. Connectivity of links in a transit domain is higher than the ones in stub domains. Thus, it is conventional to give a higher probability for the link connectivity in a transit domain than the links in a stub domain.

In the simulations, partial view of each node is composed of its one hop neighbors. For computing the delays of the links, the propagation delays is considered because the bandwidths of the links in the network are assumed to be 100 Mbit/sec and transmission delays are very small compared to the propagation delays. As the propagation delay for each link in the network, the random delays that the GT-ITM software generates are used. The messages in the network follow the shortest path computed by the Dijkstra's Algorithm [34].

4.3 Simulation Results

In this section the behavior of Stepwise Epidemic Buffering and the effects of parameters on the system performance are examined. The parameters of the scheme clearly affect the performance results. By varying the parameters, the system is driven to work more optimally.

4.3.1 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. 4.1 shows the sketch of the topology where node numbers are indicated. The long-term buffer capacity of a node is 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 stub-domains a message is buffered to the total number of bufferers of the message.

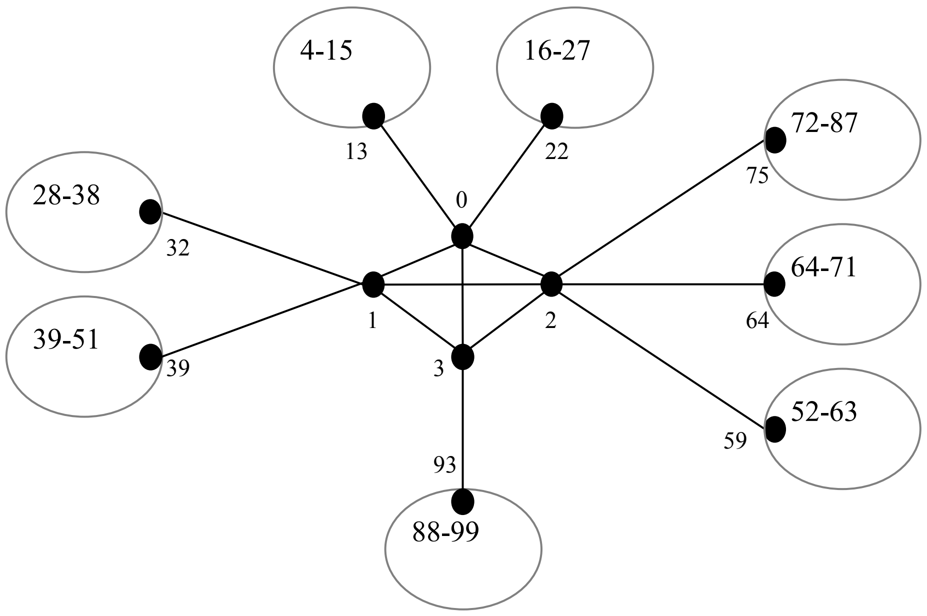


Figure 4.1 Simulation Topology

In Fig. 4.2 through 4.4, the source node is varied in terms of its position in the overlay. Fig. 4.2 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 two sets of results. The improvements are obtained by trials on the forwarding probabilities. In these trials, we aim to provide a more uniform buffering load distribution in view of the topology of Fig. 4.1. The standard deviation of the messages buffered among all nodes is given as a distinguishing metric for comparison of uniformity over all buffers. In the 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.

As a stub node, node 72 is chosen to be the source in Fig. 4.3. 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. 4.4, 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. 4.3 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.

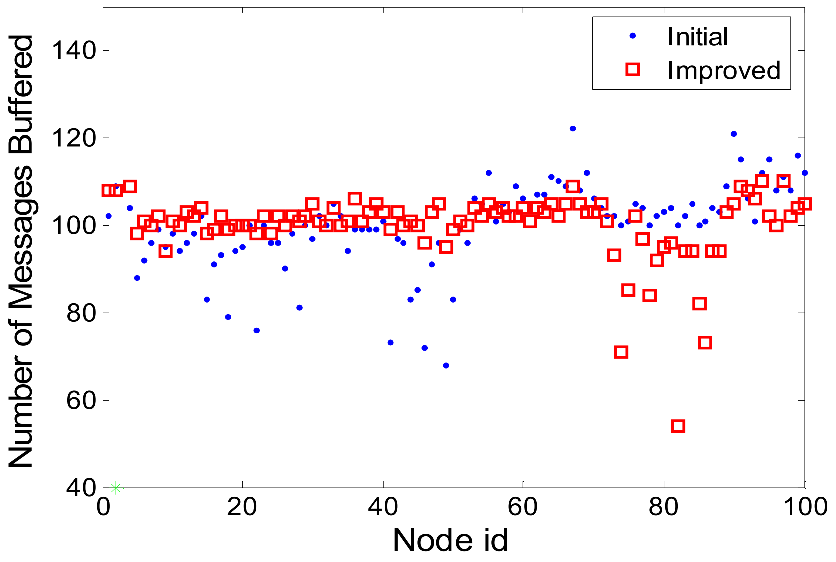


Figure 4.2 Source is a transit node

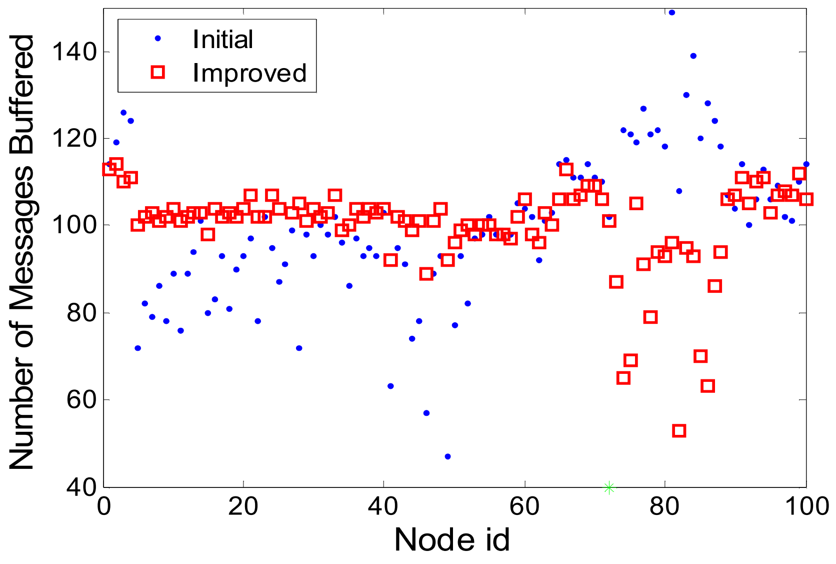


Figure 4.3 Source is a stub node

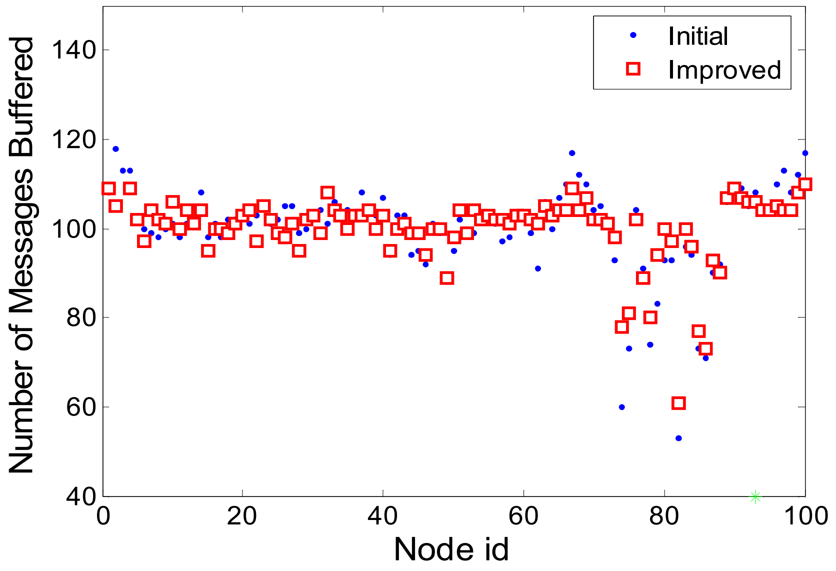


Figure 4.4 Source is stub node that has a transit neighbor

Transient behavior of the uniformity of buffer fullness by monitoring the standard deviation of the buffer levels as the message generation proceeds is investigated. 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. 4.5. When the number of generated messages approaches the full long-term buffer capacity of the system, the variability decreases which indicates a more uniform buffer load distribution.

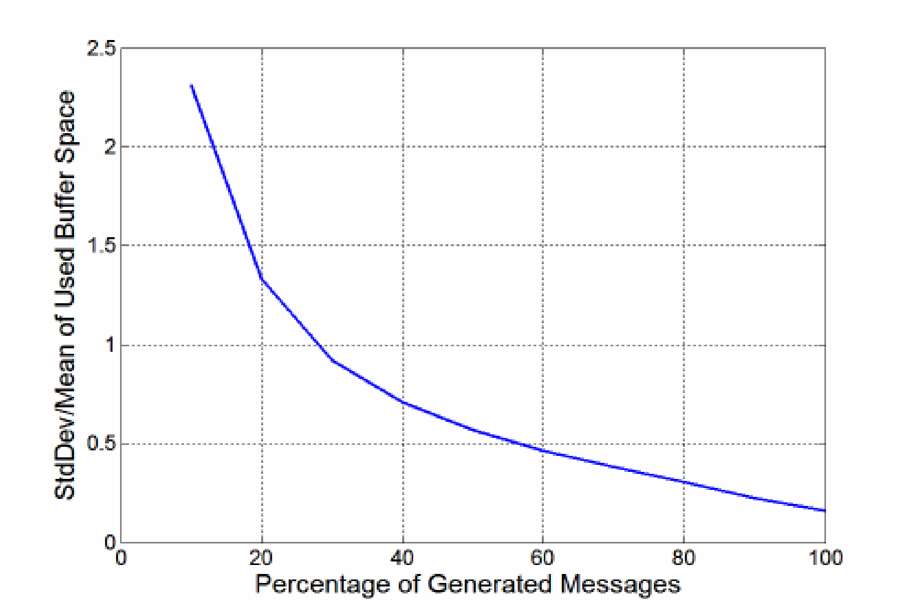


Figure 4.5 Ratio of Standard Deviation to Mean Used Buffer Space Through Message Generation

In Fig. 4.6, effect of varying the source node on retention ratio is examined. Recall that the total number of messages generated equals the total capacity of all buffers in the overlay network. Here, the total capacity of the system is the total long-term capacity of all peers. 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 this study’s scheme is robust in this respect. What is more, the retention ratio is above 97%, that is, approximately only 3% of the messages are discarded due to buffer overflows.

The effect of steps to live and the number of forwarders parameters of buffering request on retention ratio is examined in Fig. 4.7. The maximum value is obtained when STL is 40 and the number of forwarders is 35. Also it can be inferred 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. For further studies of Stepwise Probabilistic Buffering, these values can be optimized by trial and error.

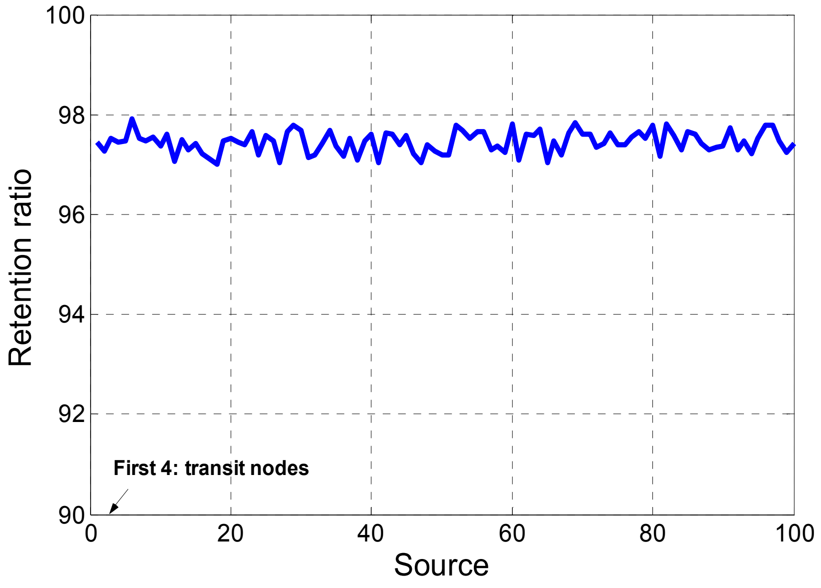


Figure 4.6 Effect of source change on the retention ratio

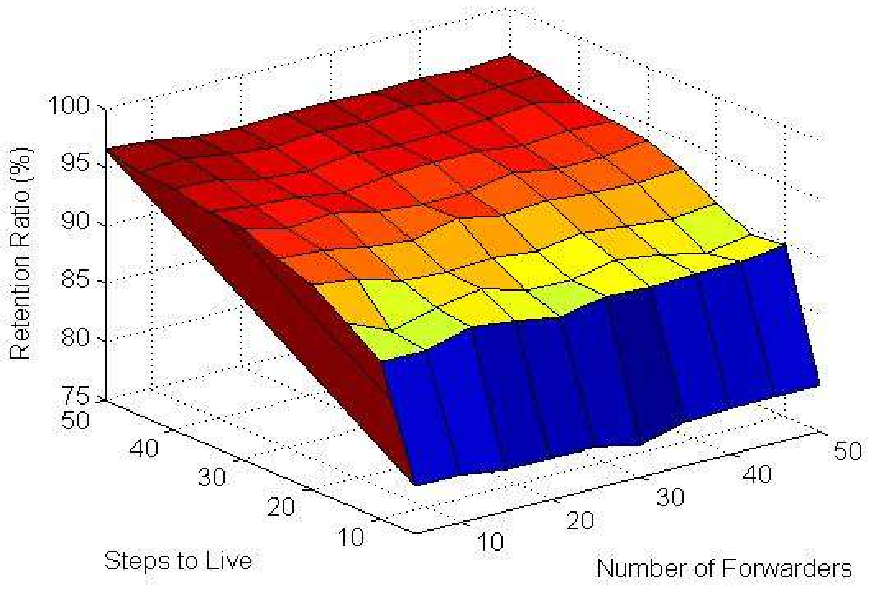


Figure 4.7 Effect of steps to live and number of forwarders on the retention ratio

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, the metric scattering ratio is used 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, 2000 messages are generated 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. It is concluded from Fig. 4.8 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.

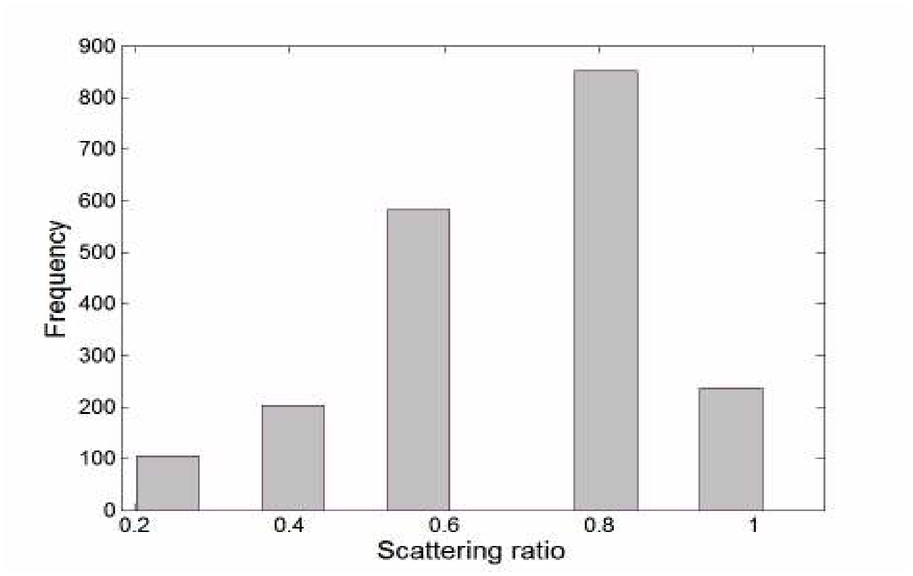


Figure 4.8 Scattering of bufferers to different domains

4.3.2 Data Dissemination and Comparative Results

Comparison of Stepwise Probabilistic Buffering with the hash-based approach [16] and our preliminary work Random Buffering [32] 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. 4.9. 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 almost as uniformly as the other approaches. In this simulation, standard deviation of buffering load is approximately 5 messages in random and hash-based approaches and 7 messages in Stepwise Probabilistic Buffering.

In the hash-based buffering [4], each message is buffered on a random subset of the membership. The subset has a desired constant size *C*. Link failures and other randomized effects in the approach can cause messages to be buffered on more or fewer than the desired member count *C*. A hash function is used to map a bitstring to a number between 0 and 1. The bitstring is formed by the message identifier and member address. A member with address A, with a view of membership of size n, buffers a message M if and only if $H(<M,A>) \times n < C$. The hash function uses a table of 256 randomly chosen integers, called the shuffle table. The algorithm is given below:

```
unsigned integer hash = 0;
for each byte b do;
   hash = hash XOR shuffle[b XOR least_signif_byte ( hash ) ];
return hash / MAX_INTEGER;
```

If a member detects that it lacks a message, it calculates the set of bufferers for the message using the hash-function and picks one bufferer at random. The member then sends a retransmission request directly to the bufferer, specifying the message identifier and the destination address. A bufferer, on receipt of such a request, determines if it has the message buffered. If so, it satisfies the request. If not, it ignores the request.

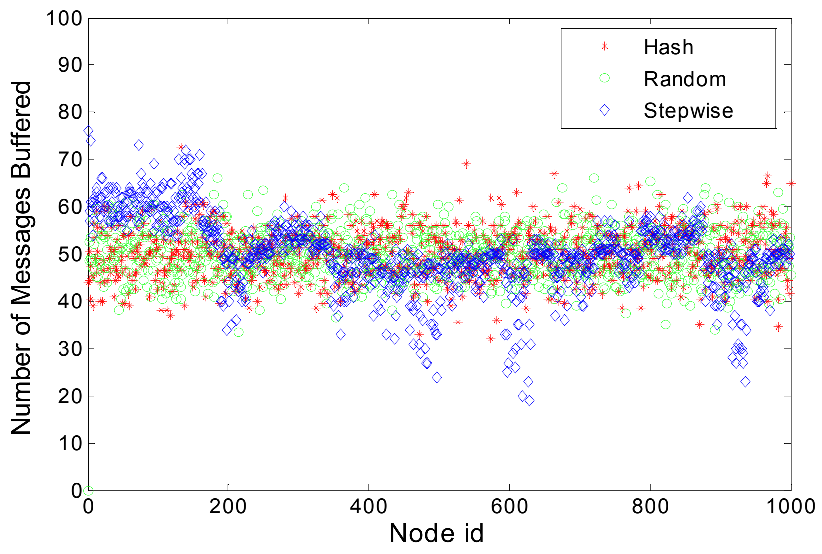


Figure 4.9 Comparison of buffering load distribution in large scale

Evaluation of the behavior of Stepwise Probabilistic Buffering in terms of message dissemination metrics as a function of parameters of the model is also done. The following performance metrics are used 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.
- *Long-term / Short-term Buffering Time* is the mean time that a message is stored on a member's long-term / short-term buffer.
- *Message Delay* is the duration between the generation of the message from the message source and the delivery of it by a receiver node.
- *Dissemination Time* is the time that passes for dissemination of the content to all peers.

Long-term and short-term buffer sizes are the main parameters that have a significant effect on the performance of Stepwise Probabilistic Buffering. When the long-term buffer size of the nodes is increased, the reliability of the dissemination is affected directly. The impact of the long-term and short-term buffer sizes on reliability is given in Fig 4.10. In these simulations, short-term and long-term buffer sizes of the nodes are raised from 2 to 20 messages. It is observed that, long-term buffer size has a positive effect on the reliability of the dissemination. Similarly, short-term buffer size of the nodes also increases the reliability but this increase occurs until a certain value of long-term buffer size. After this threshold, the reliability is almost %100.

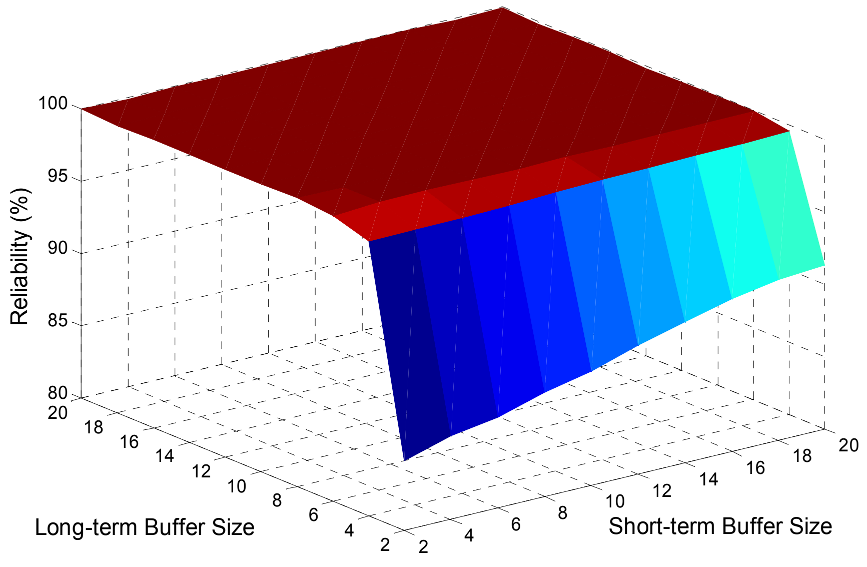


Figure 4.10 Effect of long-term buffer size and short-term buffer size on reliability

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. 4.11 shows the results of the simulations which support these facts.

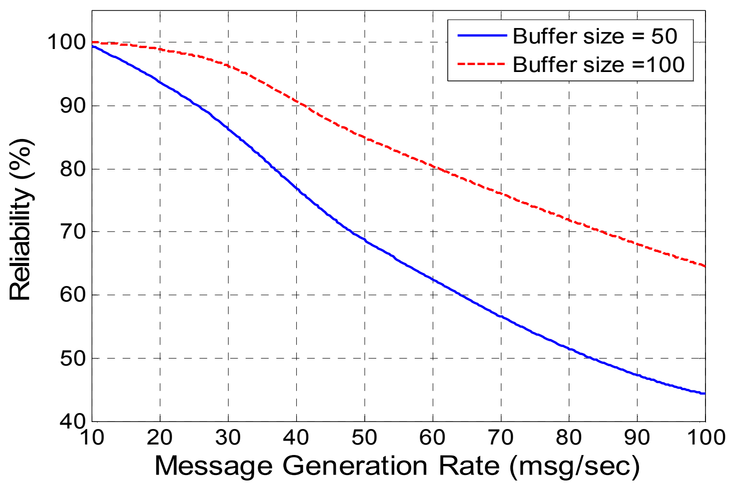


Figure 4.11: Message Generation Rate - Reliability

When the gossip interval is increased, similar effects are observed as shown in Fig. 4.12. As it is 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.

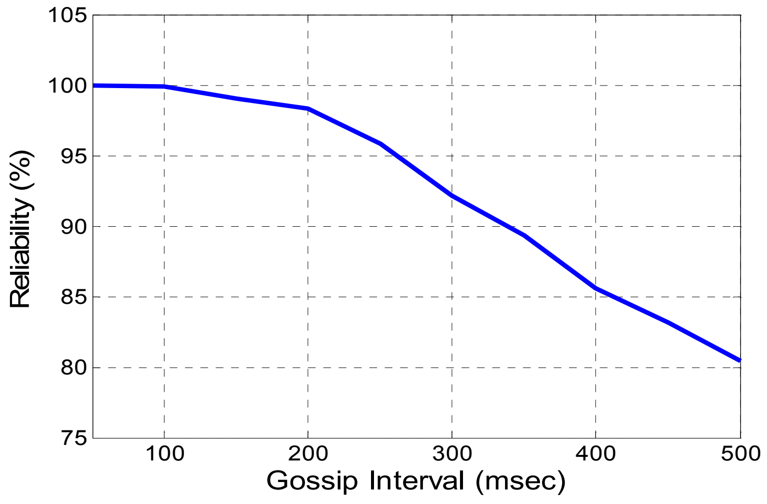


Figure 4.12 Gossip Interval – Reliability

In addition, Fig 4.13 shows average message delay as a function of message generation rate at the source. As soon as the bufferers of a message are selected, it is directly sent to the bufferers. When number of generated messages per unit time increases, the long-term buffers of the nodes would store more messages in a given time interval and a node can achieve a missed message in a shorter time. Thus, average message delay decrease as the rate of message generation grows up.

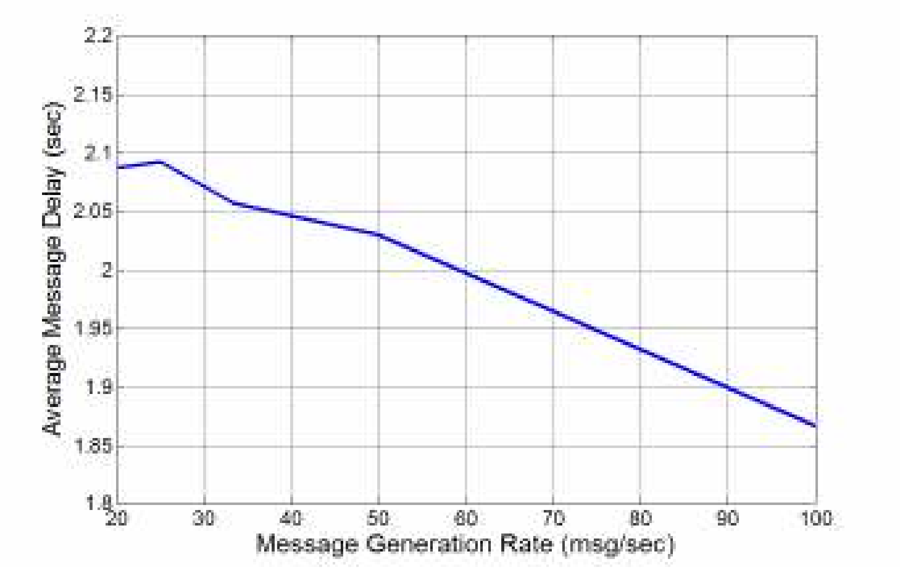


Figure 4.13 Message generation rate – Average Message Delay

In the simulations explained up to this point, the data messages are generated by a single source. The effect of number of sources on reliability of the dissemination is examined in Fig. 4.14. Each message source operates its own stepwise buffering selection mechanism concurrently with other sources. Since the message number and the source id make a unique descriptor for the buffering requests, each source node can activate its bufferer selection mechanism. When the number of sources is increased, simultaneous message generation from different network positions is triggered. Therefore, number of messages spread to the system increases. By this way probability of retrieving a missed message in the time interval that the corresponding message is achievable decreases. Thus the reliability of the system decreases as number of sources is increased.

Comparison of stepwise probabilistic model with the hash-based approach [4] and random buffering [32] in terms of dissemination time in a 1000 node scenario is given in Fig. 4.15. 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. 4.15, 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.

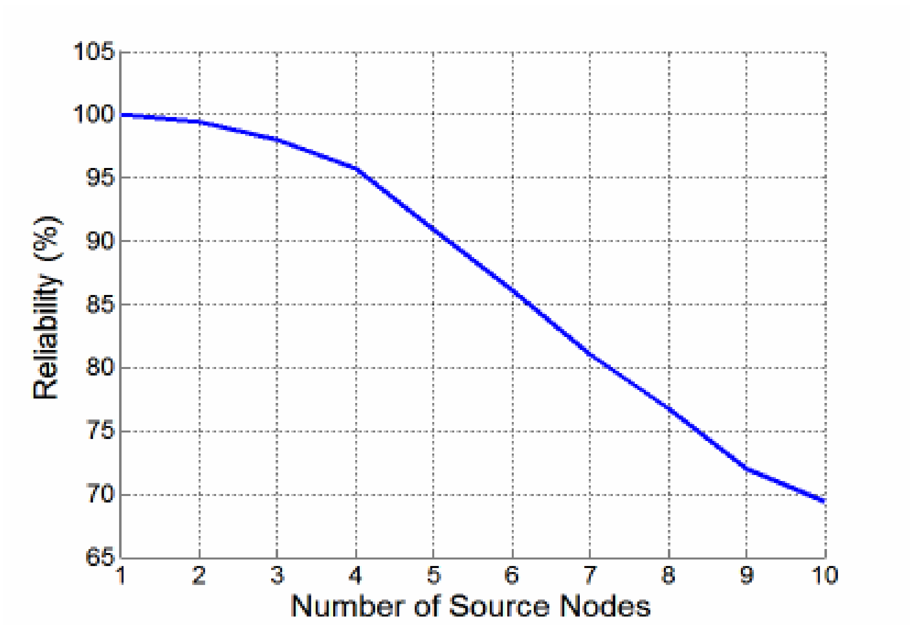


Figure 4.14 Number of Data Sources – Reliability

Comparison of the mean long-term buffering time of each peer is given in Fig. 4.16. 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.

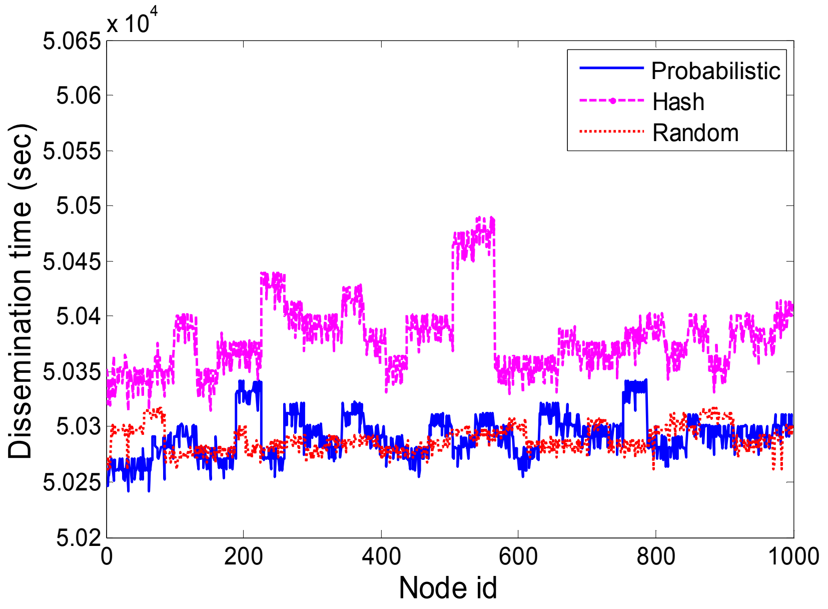


Figure 4.15: Comparison of content dissemination times

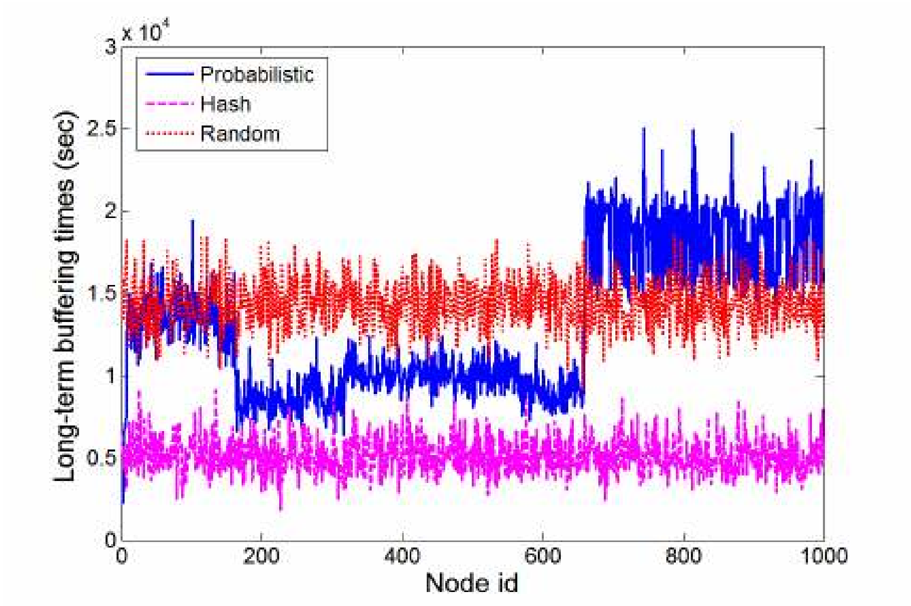


Figure 4.16: Comparison of long-term buffering times

Chapter 5

STEPWISE FAIR-SHARE BUFFERING

In Stepwise Probabilistic Buffering which is explained in detail in the last two chapters, the primary aim is to be able to choose bufferers uniformly throughout the system so that the buffering load will be well balanced among the peers where every peer has a partial view of the system. Here the buffering load is defined as the total number of messages (information unit) which are buffered by a peer during the dissemination of information. The scheme uses a probabilistic algorithm that works on every peer to determine the bufferers of a message. Every peer sends the buffering requests randomly to one of its neighbors with a probability equal to its buffer fullness ratio. The scheme provides a fairly uniform distribution in a partial view scenario.

The probabilistic algorithm works when the number of generated messages is lower than the total long-term buffer capacity of the system. When the long-term buffers of the peers become full, if a member receives a buffering request message it directly sends the buffering request to one of its neighbors and the receiving neighbor does the same process again. Therefore the buffering request is forwarded peer by peer until the *steps-to-live* value expires. The nodes on which the *STL* value expires would buffer the corresponding message in that case.

The uniformity of buffering load distribution of the probabilistic algorithm is observed only when the number of generated messages approaches the total long-term buffer capacity of the system. The algorithm does not provide uniformity before the nodes reach their long-term buffer sizes similar to the case when the number of generated messages exceeds the total long-term buffer capacity. In the scheme, initially, assuming that all buffers are empty, peers that are in the partial view of the source will accept the buffering requests with higher probability than the other ones. Thus, if a snapshot of the system is taken when the number of generated messages is equal to a small fraction of the total long-term buffer capacity, a large deviation is observed on the buffering load of the peers. Specifically, the buffer levels of the nodes close to the source node are much higher and the buffer levels of the ones far from the source are approximately zero.

The improvements of the scheme help to balance the buffering load among peers but they lead to an overhead on the buffering request message and the computations become infeasible for a large scale network. Besides, these optimizations are specific to the given network topology. When the algorithm is deployed on another network topology the necessary computations must be done particular to that topology. Since the ids of the nodes forwarding the buffering request are also included; the size of the buffering request message increases and this gives rise to an overhead on the network. For the other optimization given in section 3.5, some specific forwarding probabilities must be computed for the network. In the numerical experiments of the Stepwise Probabilistic Buffering scheme, these probabilities are computed for a small scale particular network topology. For a large scale scenario, the computations of the probabilities are infeasible.

A more robust scheme Stepwise Fair-share Buffering is developed to remove the inconveniences of the probabilistic algorithm. Stepwise Fair-share Buffering provides more uniform load distribution. It is scalable, simple and applicable to any kind of underlying network topology. It does not bring an additional overhead on the buffering request message. In this chapter, the details of the fair-share scheme are given.

5.1 Fair-share Buffering Algorithm

The Stepwise Fair-share Buffering is designed as an improved and robust version of Stepwise Probabilistic Buffering scheme. It is based on assumptions and is developed for epidemic information dissemination as the probabilistic approach. Every peer has partial membership knowledge and anti-entropy model is used for dissemination. The two phase buffering mechanism is used also in this scheme.

In the method, every peer stores the number of messages that its neighbors have ever buffered. This is called the *neighbor history* information (NH). This information is used for determination of the bufferers. At specific time intervals, the peers update their neighbor history information. The bufferer determination phase is initiated by the source to one of its neighbors through a selection mechanism. 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 it decreases the *STL* value attached to a buffering request message. If the *STL* value becomes zero, then the peer accepts the buffering request. If *STL* value is greater than zero, the peer multicasts *neighbor history request messages* to its neighbors. As soon as the peer receives all the responses from the neighbors, it updates its *neighbor history* information. Then, it detects the peers with the minimum number of messages buffered. If the corresponding peer is the peer itself it accepts the buffering request, otherwise if it is one of the neighboring peers it sends the buffering request to that neighbor. If there is more than one peer with the minimum number of buffered messages, the peer chooses randomly one of them. Similarly, if the peer is one of these candidate peers and it chooses itself then it accepts the request. Fig. 5.1 shows the steps of the fair-share algorithm. There is no last forwarder mechanism or forwarding probabilities in this scheme. Only, the receiving node does not send the buffering request message to the peer that it has received the message from.

5.2 Improvements

To ensure that the algorithm works in all cases, we made some certain improvements. In this section, we describe the details of these improvements.

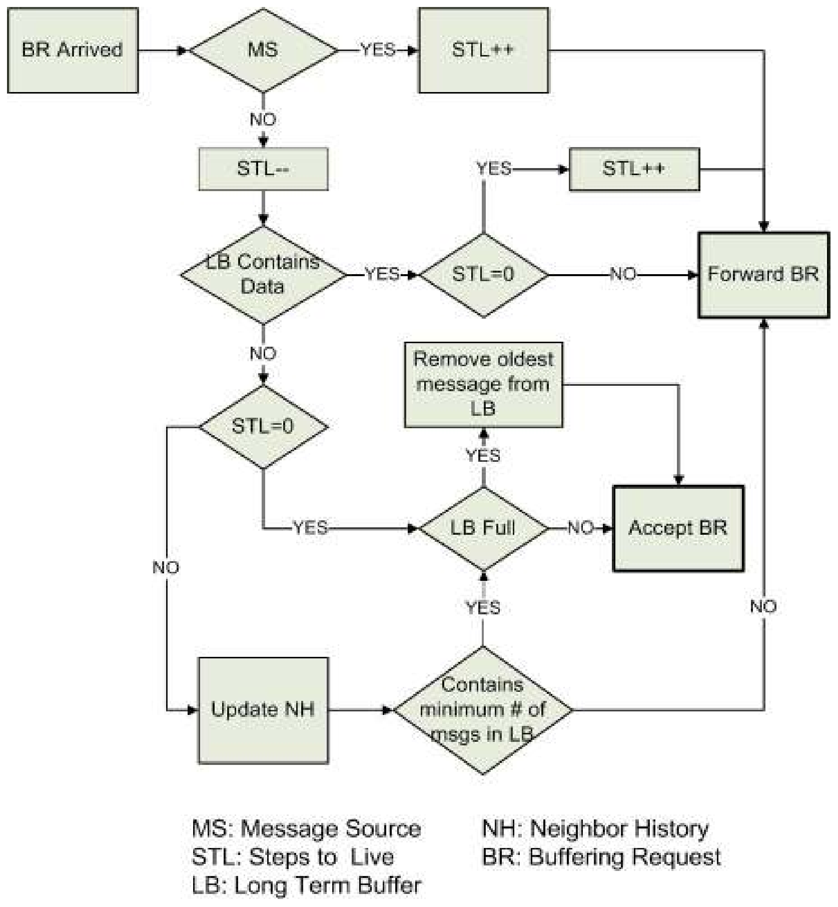


Figure 5.1 Flow chart of the fair-share algorithm

5.2.1. Handling Fast Request Rate

When a buffering request is received by a peer, it multicasts *neighbor history request* messages to its neighbors as explained above. Then, a certain time passes until all the response messages are received by the peer. Therefore, a key point in the mechanism is to make adjustments when the rate of receiving a buffering request message is faster than the rate of updating the neighbor history. In this case, before the peer receives the responses from its neighbors, more than one buffering requests accumulate in the *buffering request list (BL)* of the peer. When the peer updates its neighbor history, it employs the same algorithm and updates its neighbor history for each buffering request in the *buffering request list* one by one. This mechanism balances the load in case of faster reception of buffering requests.

Let us illustrate this idea with a simple scenario. In the network topology given in Fig. 5.2 a) and b), peer 1 has three neighbors. Peer 2, 3 and 4 has 2, 3 and 5 messages in their long-term buffers respectively as it is indicated in Fig 5.2.a). Assume that until peer 1 receives the response messages from its neighbors, five messages accumulate in its *buffering request list*. Then, it sends message 1 to peer 2 because it has the minimum number of messages. As a result, the number of messages that peer 2 and 3 have in buffer become equal to 3. After that, peer 1 selects randomly peer 3 and sends message 2 to it. Then, it sends message 3 to peer 2, message 4 to peer 3 and message 5 to peer 2 as it can be observed from Fig. 5.2.b). Consequently, every peer receives 5 buffering request messages and the load is distributed evenly.

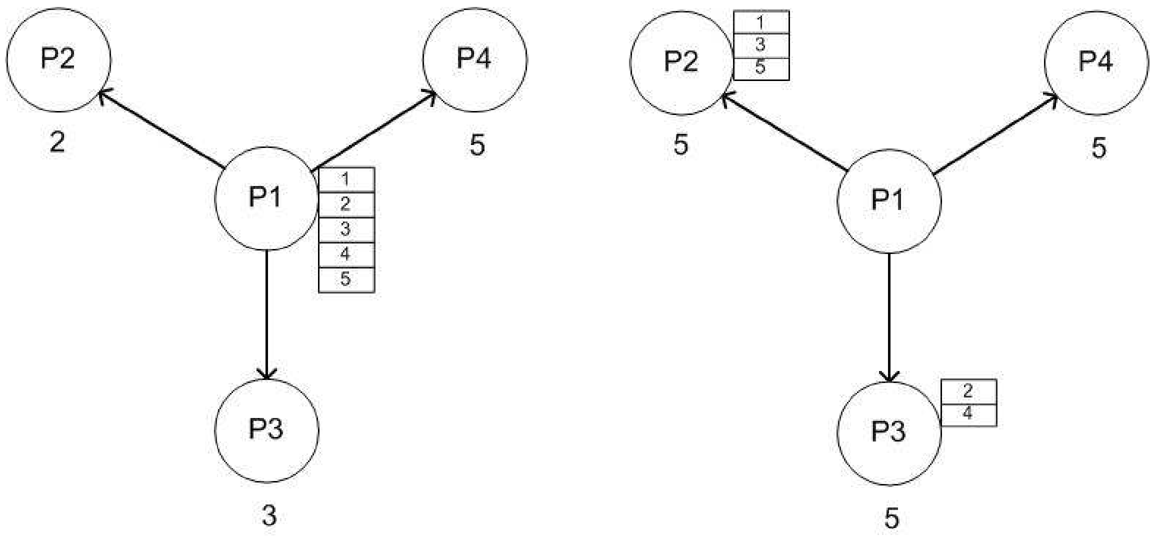


Figure 5.2 Handling Fast Request Rate: Example Scenario
a) 5 request messages accumulated b) Load is distributed evenly

5.2.2. Timeout Mechanism

Every peer waits for the response messages before it detects the peer with the minimum number of messages. If there is a link failure on one of the links that is attached to a neighbor, or one of the neighbors crashes, the peer should wait forever.

Not to lead to such a situation, a timeout mechanism is developed. For this purpose, in the scheme a peer waits for the response messages up to a *timeout value T*. This time-out value *T* can be set greater than the maximum round trip time among the neighbors of the peer.

5.3. Events, Data Structures and Variables

In this section the events, data structures and variables of the Stepwise Fair-share Buffering mechanism are described. The events, data structures and variables indicated with a * are specific to Stepwise Fair-share Buffering. The others are the same with the Stepwise Probabilistic Buffering. The events are given in table 5.1, the data structures are given in table 5.2 and the variables are given in table 5.3 with the corresponding descriptions.

Table 5.1. The events

Event	**Description**
Data Reception	Reception of a data message
Request Message Reception	Reception of a request message
**Buffering Request Reception*	Reception of a buffering request message
**Neighbor History Request Reception*	Reception of a neighbor history request message
Gossip Propagation	Gossip dissemination procedure operated periodically
Digest Message Reception	Reception of a digest message of a neighboring node
Long-term Buffer Insertion	Insertion of a data message into the long-term buffer
Short-term Buffer Insertion	Insertion of a data message into the short-term buffer
Data Generation	Generation of data by the source node
Bufferer Selection	Selection of bufferer ids of a data message

Table 5.2. The data structures

Data Structure	Description
DATA_MESSAGE	Data message received
REQUEST_MESSAGE	Request message to be send for data reception
BUFFERING_REQUEST	Buffering request message sent by the source node
**NEIGHBOR_HISTORY*	Number of messages that the neighbors buffered
**BUFFERING_REQUEST_LIST*	List of buffering requests received by the peer
**MIN_PEERS*	List of peers with the minimum number of buffered messages
MESSAGE_ENTRY	Message id and bufferers of the messages
RECEIVED_MESSAGES	Message entries of received messages
ACCEPTED_REQUESTS	List of buffering requests accepted for buffering
NEIGHBOR_LIST	List of neighboring nodes in the partial view of current node
DIGEST_MESSAGE	Digest message to be sent

Table 5.3. The special variables

Variable	Description
Message_Id	Unique id of each data message
Gossip_Round	Gossip round counter increased in each gossip round
Fan-out	Number of nodes chosen for gossiping each gossip round
Number_Of_Bufferers	Number of bufferer nodes for the data messages
Generation_Interval	Time interval of data generation determined by the source node
Digest_Size	Number of entries in the digest message
Long-term_Buffer_Capacity	Number of messages that can be stored in the long-term buffer
Short-term_Buffer_Capacity	Number of messages that can be stored in the short-term buffer
**Timeout*	Waiting time until all response messages received
Steps_To_Live	Max number of hops the a buffering request can travel
STL_Counter	Remaining lifetime of buffering request as number of hops

There are three message types special to the Stepwise Fair-Share Buffering. These are buffering request message, neighbor history request message and response message. Formats of these messages are provided in table 5.4.

Table 5.4. Message formats

BUFFERING_REQUEST_MESSAGE:				
Message Type *1 octet*	Size of Message 2 octets	Message Id 1 octet	STL value 1 octet	Source Id 1 octet
NEIGHBOR_HISTORY_REQUEST_MESSAGE:				
Message Type *1 octet*	Size of Message 2 octets	Sender Id 1 octet		
RESPONSE_MESSAGE:				
Message Type *1 octet*	Size of Message 2 octets	Message Id 1 octet	Number of Buffered Messages Max 1024 octets	Source Id 1 octet

5.5. Algorithms of the Events for Stepwise Fair-share Buffering

In this section, algorithms for bufferer determination phase are given. The algorithms for the data dissemination phase are the same as Stepwise Probabilistic Buffering. The following are the algorithms special to the buffering request reception and neighbor history request message reception events of Stepwise Fair-Share Buffering.

Buffering Request Reception:

```
If (Is_Message_Source (BUFFERING_REQUEST)) then;
  Increase the STL_Counter of BUFFERING_REQUEST
  Buffering Request Transmission (BUFFERING_REQUEST)
Else
  Decrease the STL_Counter of BUFFERING_REQUEST
  If (LONG-TERM_BUFFER contains DATA_MESSAGE) then;
    If (STL_Counter=0) then;
      Increase the STL_Counter
    Else
      Buffering Request Transmission (BUFFERING_REQUEST)
    Endif
  Else
    If (STL_Counter=0) then;
      If (size of the LONG_TERM_BUFFER = long-term_buffer_capacity) then;
         Remove the last DATA_MESSAGE from LONG_TERM_BUFFER
      Endif
      Add the BUFFERING_REQUEST to ACCEPTED_REQUESTS
    Else
      Multicast NEIGHBOR_HISTORY_UPDATE_MESSAGE to peers in the NEIGHBOR_LIST
      If (all RESPONSE_MESSAGES are received) then;
        For all BUFFERING_REQUESTs in the BUFFERING_REQUEST_LIST;
          Determine peers with min number of buffered messages
          Choose a peer randomly from MIN_PEERS
          If (Min_peer) then;
            Add the BUFFERING_REQUEST to ACCEPTED_REQUESTS
          Else
            Buffering Request Transmission (BUFFERING_REQUEST)
          Endif
        Endfor
      Else
        Wait for the Timeout
        For all BUFFERING_REQUESTs in the BUFFERING_REQUEST_LIST;
          Determine peers with min number of buffered messages
          Choose a peer randomly from MIN_PEERS
          If (Min_peer) then;
            Add the BUFFERING_REQUEST to ACCEPTED_REQUESTS
          Else
            Buffering Request Transmission (BUFFERING_REQUEST)
          Endif
        Endfor
      Endif
Endif
```

Neighbor History Request Message Reception:

```
Determine the id of the sender from NEIGHBOR_HISTORY_REQUEST_MESSAGE
Piggyback the number of messages in the long-term buffer to the RESPONSE_MESSAGE
Send the the RESPONSE_MESSAGE to the sender
```

Chapter 6

SIMULATION MODEL AND ANALYSIS OF STEPWISE FAIR-SHARE BUFFERING

To evaluate the performance of Stepwise Fair-Share Buffering scheme, the simulation platform explained in Chapter 4 is used. In this chapter, first the experimental results of Stepwise Fair-share Buffering and its comparison with the Stepwise Probabilistic Buffering in terms of distributing the buffering load are exhibited. After that, evaluation of the model in terms of data dissemination metrics and comparison with Hash-based [4], and Stepwise Probabilistic Buffering [17], Random [32] schemes are given.

6.1 Uniform Bufferer Selection

We evaluate the performance of Stepwise Fair-share Buffering in terms of distributing the buffering load. In the first group of experiments, a 100-node transit-stub topology is used. Long-term buffer capacity of the nodes is 100 messages, and 10,000 messages are disseminated from a single source. This value is equal to the total long-term buffer capacity of the nodes. STL value is set to 20 which was 30 in the experiments of Stepwise Probabilistic Buffering. Each message is buffered only by a single bufferer in our simulations.

In Fig. 6.1-6.3 similar to Fig. 4.3-4.5, the source node is varied in terms of its position in the overlay. The source (represented by a diamond on the *x*-axis) is a transit node (id-2) in Fig. 6.2, a stub node (id-72) in Fig. 6.3 and an intermediate node (id-93) in Fig. 6.4. An intermediate node connects a transit node to a stub domain. In comparison to stepwise probabilistic buffering, the uniformity of the distribution of buffering load is significantly higher in the fair-share scheme. When the location of the source node on the topology is varied, namely source is positioned on a transit or a stub node, we have observed similar results. Besides, the distribution behaves the same for all the stub-domains. For example, the load of the nodes close to the source node is not higher or lower in contrast to the stepwise probabilistic approach. The comparison of numerical values of the standard deviations of buffering load for three types of sources is given in Table 6.1. These values also show the significant difference in the standard deviations of the two approaches.

Table 6.1. Standard deviation of buffering load in figures 6.1-6.3

Source	**Std. Dev. in Probabilistic**	**Std. Dev. In Fair-share**
Transit source	7.78	1.10
Stub source	10.32	1.07
Intermediate source	8.03	0.92

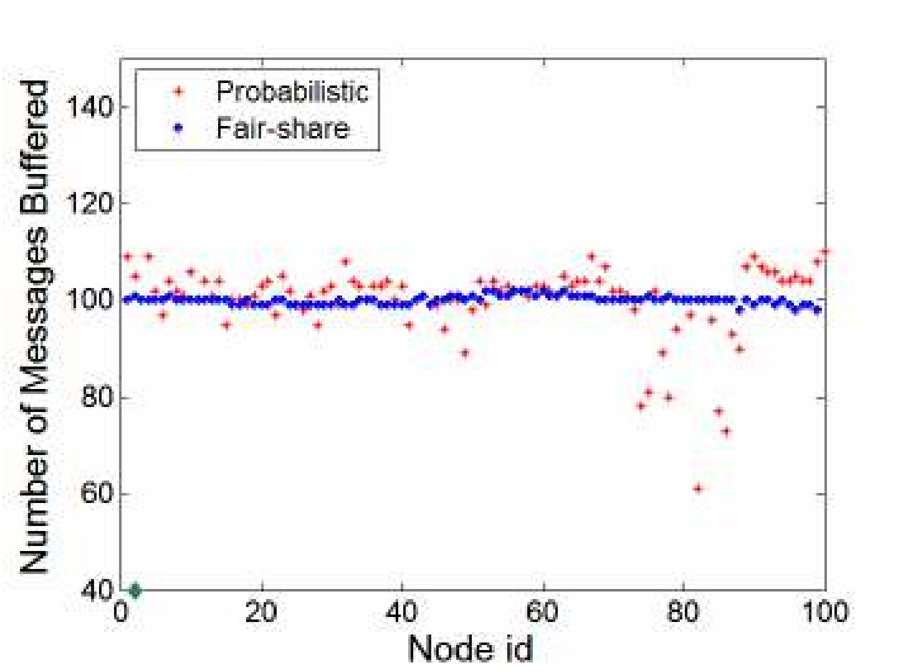


Figure 6.1 Comparison of buffering load (Source is a transit node)

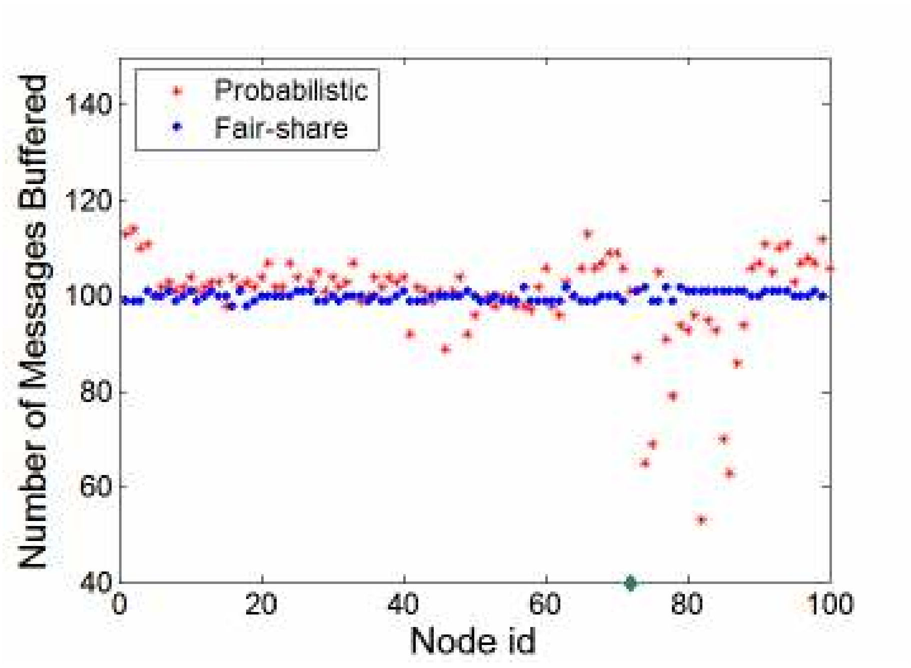


Figure 6.2 Comparison of buffering load (Source is a stub node)

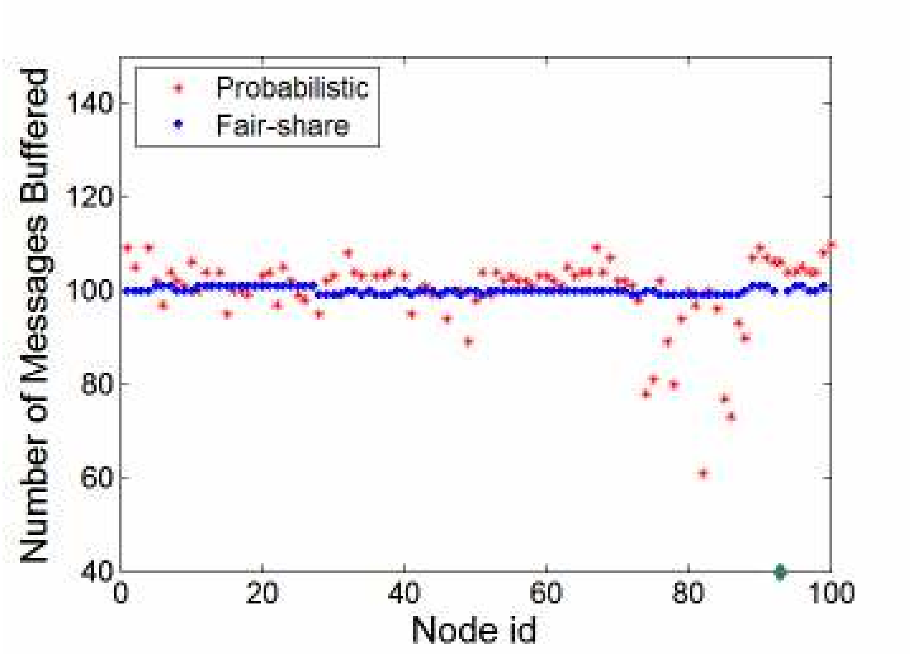


Figure 6.3 Comparison of buffering load (Source is an intermediate node)

Transient behavior of the uniformity of the buffer levels also has a great importance for dissemination. Standard deviation of the buffer levels as the message generation proceeds is monitored to investigate the transient behavior of the uniformity of buffer fullness. 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 time. As it can be seen from Fig. 6.4 the variability decreases dramatically in the Stepwise Probabilistic Buffering scheme. However, there is not such a big difference in the deviation of buffer levels in the fair-share scheme and the variability is low throughout. In Stepwise Fair-share Buffering, since the ratio of the standard deviation to the mean of the used buffer spaces of the nodes does not change considerably, the distribution of the buffering load during dissemination process is also uniform.

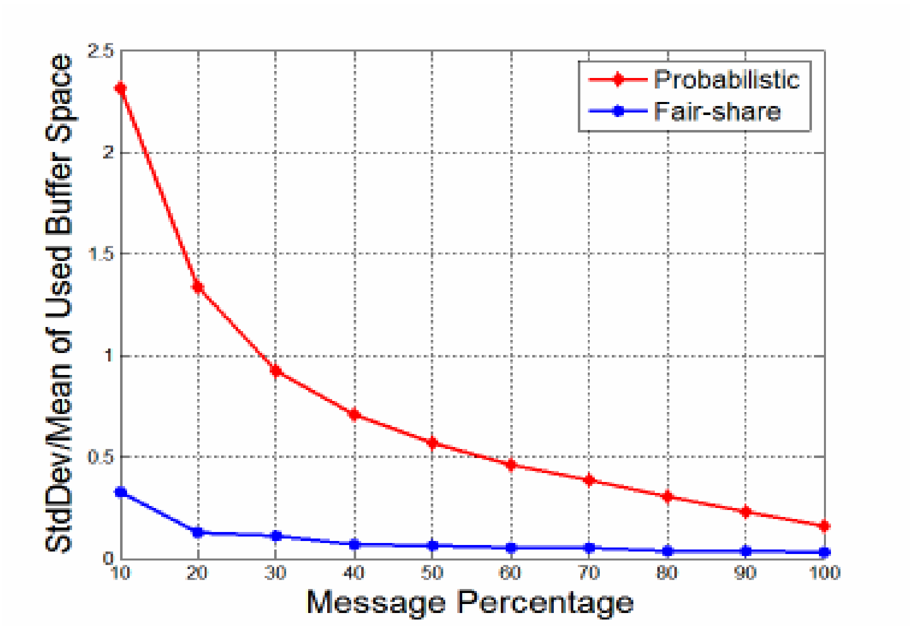


Figure 6.4 Comparison of standard deviation over mean used buffer space

In Fig. 6.5, buffering load of the nodes is given for various dissemination percentages. Long-term buffer capacity of the nodes is equal to 50 messages on a 100-node network topology as shown in the figure; the scheme provides stability of uniformness over time which will be helpful for message dissemination.

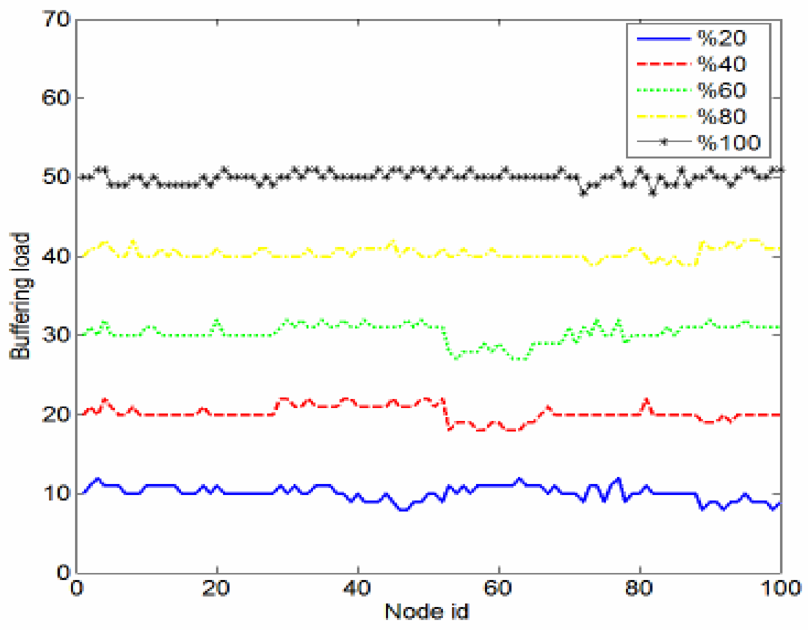


Figure 6.5 Uniformity of the fair-share scheme in time

Fig. 6.6 shows the resultant retention ratio for different source nodes. Recall that, retention ratio is the ratio of the number of messages retained in all long-term buffers to the total number of messages generated. In each simulation step, one individual node is chosen as the source and the retention ratio is calculated for that particular source. This simulation is repeated for every individual node in the system. The figure shows that the retention ratio does not change significantly as the source changes for both approaches. It can also be inferred that the retention ratio in the fair-share approach is higher than the probabilistic one. This result is due to more uniform buffering load distribution in the fair-share scheme.

Comparison of buffering load distribution on a 1000 node topology is given in Fig. 6.7. In this network topology, there are 10 transit nodes. Each transit node is connected to 11 stub domains and each stub domain contains 99 nodes on the average. In these simulations, forwarding probabilities for the Stepwise Probabilistic Buffering are optimized by trial and error. When the scale of the system increases determining these probabilities becomes hard. In the probabilistic approach, the bufferers are determined according to the buffer fullness level and the computed forwarding probabilities. As the scale of the network increases, the uniformity disappears. But, in the fair-share scheme the data is distributed uniformly for a large scaled system using its unique idea explained in Chapter 5.

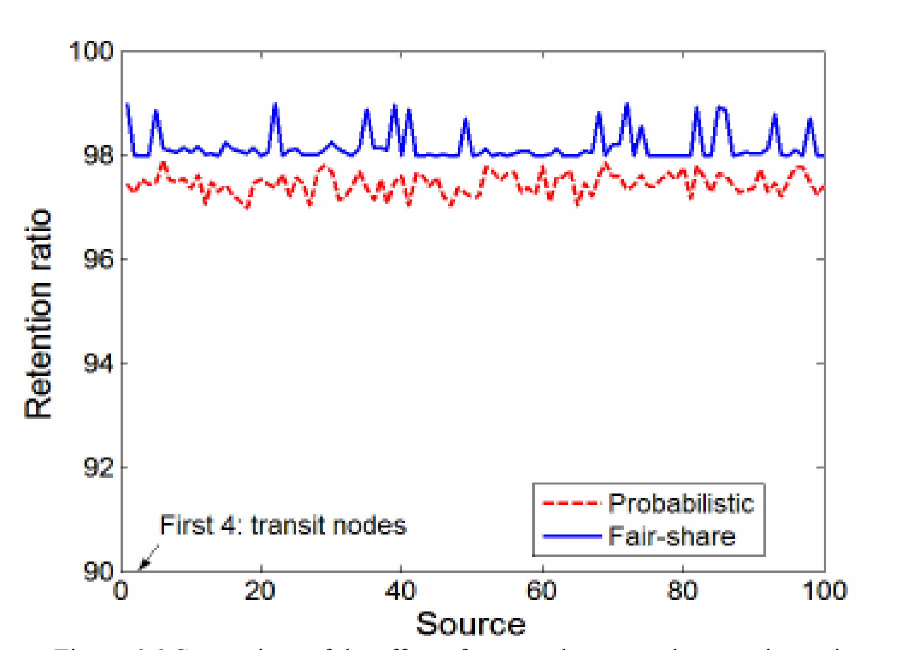


Figure 6.6 Comparison of the effect of source change on the retention ratio

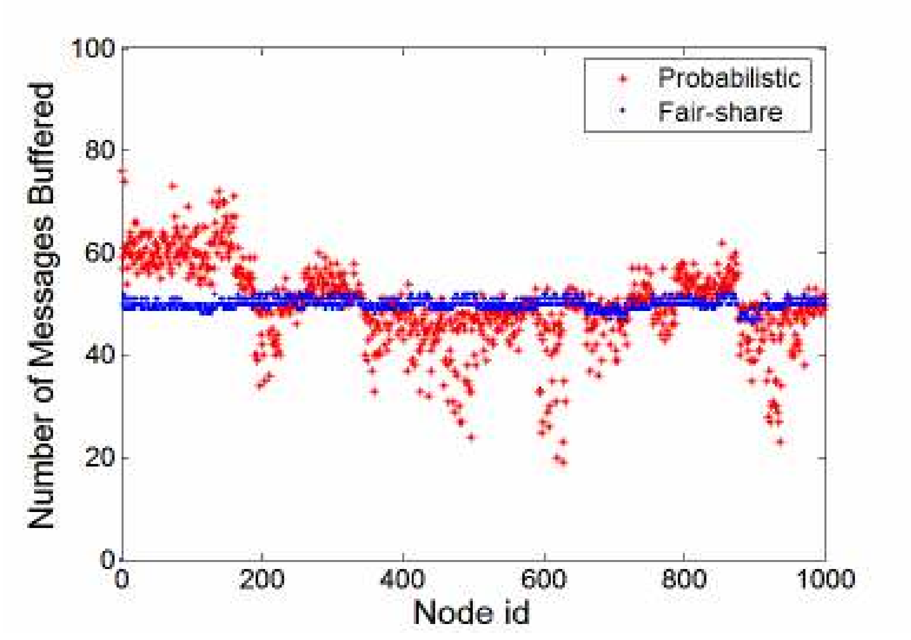


Figure 6.7 Comparison of buffering load distribution in large scale (1000 node)

6.2 Data Dissemination

In this section, the evaluation of Stepwise Fair-share Buffering and its comparison with other approaches are given in terms of data dissemination metrics. The metrics investigated are reliability, content dissemination time, buffering times, message delay and the minimum buffer requirement of the system.

In Fig 6.8, the minimum buffer requirement per node is investigated as the scale of the system increases from 500 to 2000 peers. The simulations are done on the transit-stub topologies of various sizes and the other parameters are kept the same. In these simulations, short-term buffer size per node is zero, that is, only the long-term buffer is used. The message generation rate is 100 messages/sec and the gossip interval is 200 msec. Analysis results show that the minimum buffer requirement decreases as the system size scales up. Since the number of nodes increases, the rate of being bufferer per node decreases and the waiting time of a message in the buffer increases. Thus, smaller buffer sizes begin to be sufficient for a message to be delivered by all members if the size of the network gets larger.

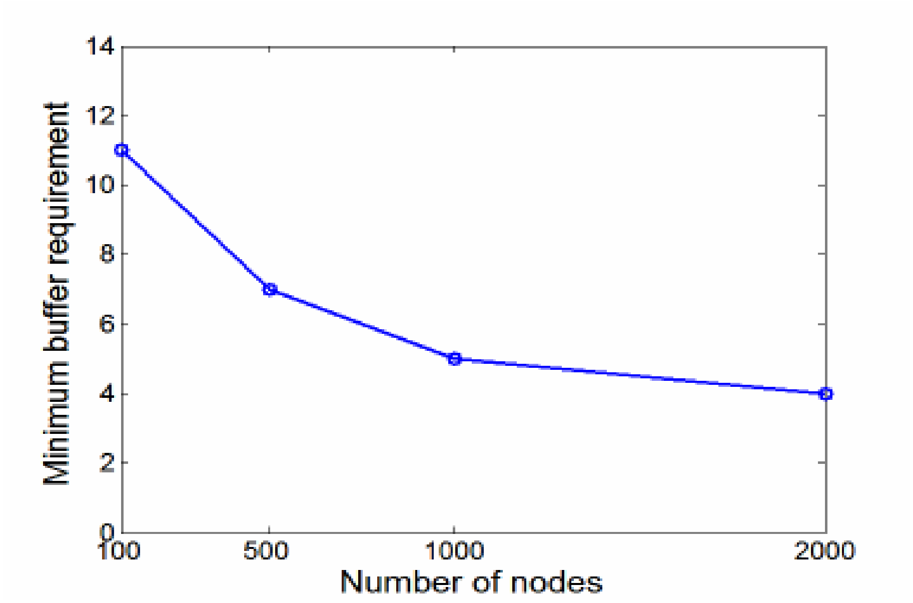


Figure 6.8 Minimum buffer requirements for a reliable dissemination in small to large scale networks

In consistency with Fig. 6.8, the results in Fig. 6.9 indicate the reliability of the data dissemination as a function of the long-term buffer size. The system achieves full reliability for the buffer sizes labeled as the minimum buffer size in Fig. 6.8.

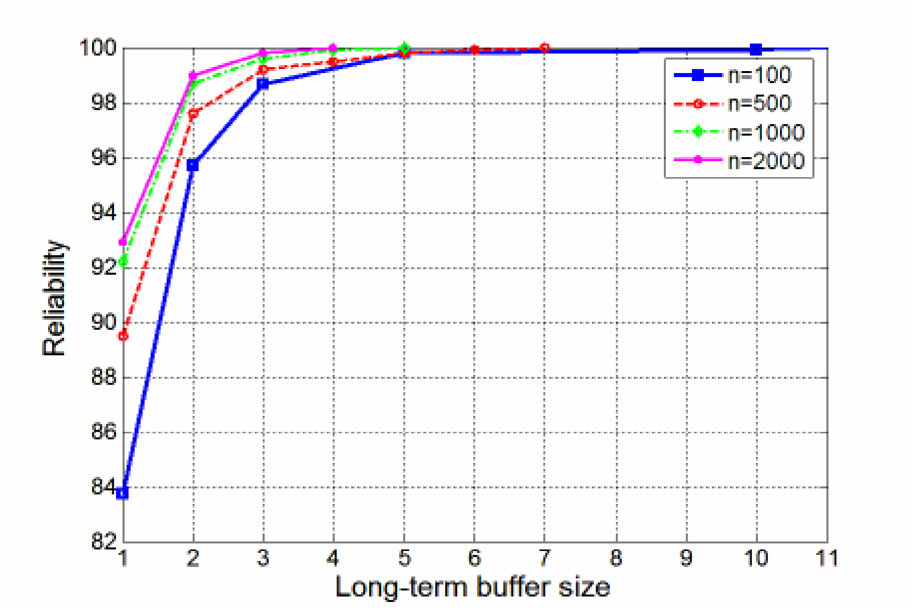


Figure 6.9 Reliability as a function of long-term buffer size in like Fig. 6.8 scale networks

Stepwise Fair-share scheme is compared with the hash-based approach [4], Stepwise Probabilistic Buffering [17] as well as a third approach called random buffering given in [32]. Random buffering is used as a basis for comparison because it assumes full membership information on the source side. Therefore, the buffer selection occurs at once as well as being very uniform due to completely random selection of the bufferers. The dissemination times in a 1000 node scenario are given in Fig. 6.10. 50,000 messages are generated from a single source and the message generation rate is 20 messages /sec, so that all messages are generated in 2500 sec. The gossip interval is set to 200 msec. In the random and hash-based buffering methods, every peer has the full view of the system. As inferred from Fig. 6.10, dissemination time of Stepwise Fair-share Buffering is close to that of random buffering even though in the first scheme every peer has only partial membership information. On the other hand, Stepwise Fair-share Buffering has a lower dissemination time than the hash-based approach because in Fair-share 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. As expected, random approach has a better dissemination time than Fair-share because the bufferers are selected at random immediately. There is no significant difference between the Probabilistic buffering and Fair-share buffering in terms of dissemination time. Basically, the last message is sent out from the source at time 2500 sec., and is received by all nodes in the next few gossip rounds for both approaches.

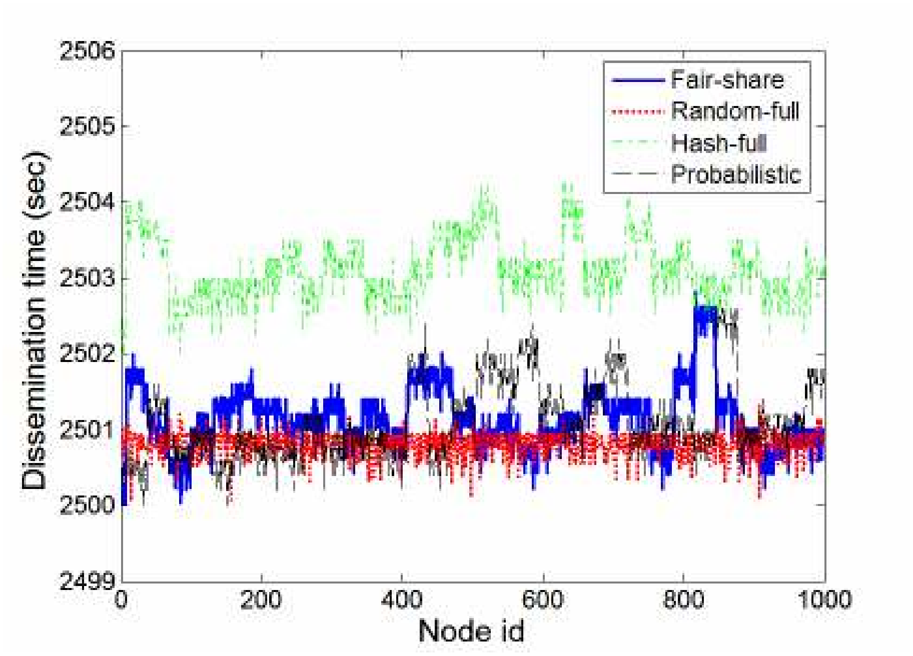


Figure 6.10. Comparison of content dissemination times

Fig. 6.11 gives the average long-term buffering time of a message in each peer for the three approaches. Average long-term buffering time of a message in random, probabilistic and fair-share is equal on the average and it is smaller in the hash-based approach. As before, the reason is that in the hash-based scheme, the messages are to be buffered eventually, but in the random and fair-share methods, the bufferers are determined at the beginning of the dissemination of a message. Long-term buffering times of random, probabilistic and fair-share approaches close to each other on the average but random and stepwise show more variability. Uniform buffering load distribution of Fair-share provides a uniform long-term buffering time distribution also. On the other hand, buffering times of physically close nodes are close to each other in probabilistic approach. This is due to the fact that the distribution of buffering load differs among the stub domains in this approach.

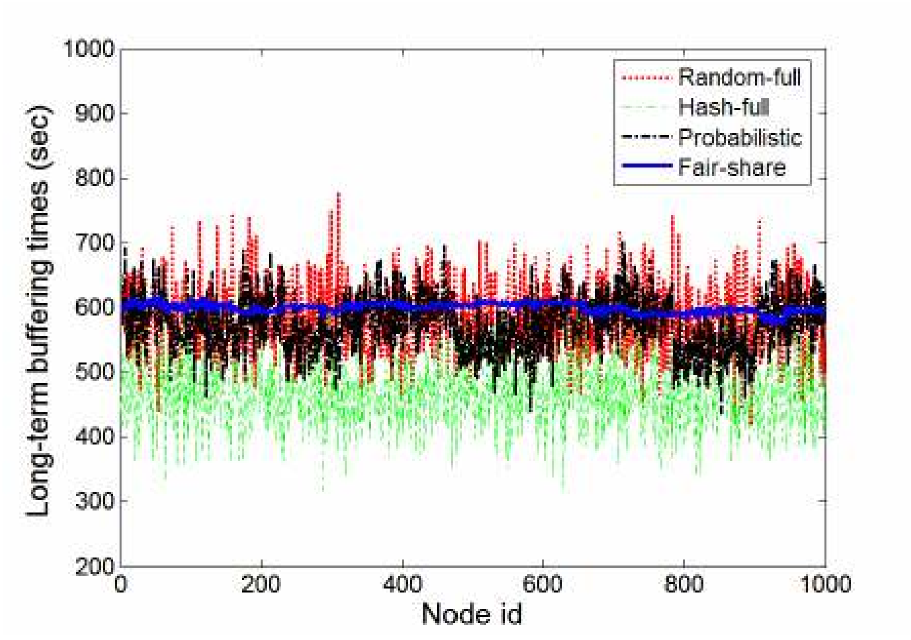


Figure 6.11 Comparison of long-term buffering times

Comparison of the average message delays on 1000-node topology simulations is given in Fig. 6.12. Stepwise Fair-share and Probabilistic buffering approaches lead to slightly higher average message delays per node, in comparison to Hash-based and Random buffering. This is due to the fact that the former approaches use additional time to determine the bufferer of each data message disseminated. However, when distributing a large content consisting of thousands of messages, bufferer determination and message dissemination phases take place concurrently, and total dissemination time for the content is not affected adversely as discussed for the results of Fig.6.10. Besides, there is no need to have full membership information in Stepwise Fair-share as well as probabilistic buffering at the expense of only slightly higher average message delays.

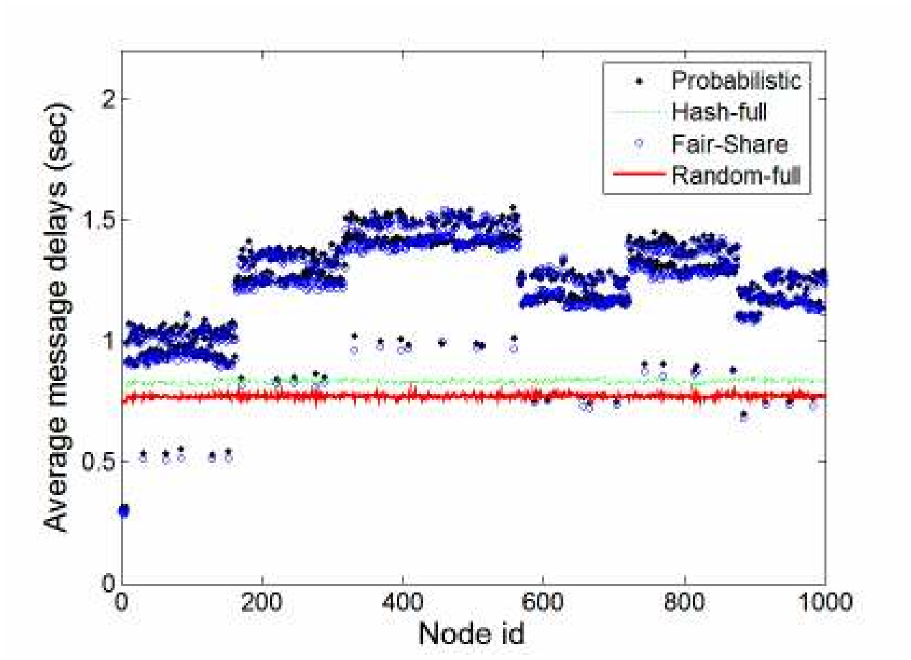


Figure 6.12 Comparison of average message delays

6.3 Scalability Results

In this group of results, the performance of Stepwise Fair-share Buffering is observed as the system size scales up. In the simulations of Fig. 6.13 to 6.16, number of nodes is increased from 1000 to 10000. Number of transit nodes and number of stub domains for each topology are provided in Table 6.2. Average number of stub nodes in each domain is 30. In these simulations, the message generation rate is 100 messages /sec, gossip interval is 200 msec and 500000 messages are disseminated to whole network.

In Fig. 6.13, the minimum buffer requirement per node is given for increasing scale. This figure is an extended version of Fig 6.8 and it can be observed that the minimum buffer requirement decreases as the group size increases. In Fig. 6.14, the standard deviation of buffering load namely the standard deviation of number of messages buffered is investigated. As it can be deduced from the figure, the standard deviation of buffering load does not increase significantly as the scale of the system increases.

Table 6.2. Node distribution for the simulation topologies

Number of nodes	Number of transit nodes	Number of stub domains
1000	8	32
2000	20	60
3000	24	96
4000	32	128
5000	40	160
6000	60	180
7000	56	224
8000	64	256
9000	90	270
10000	80	320

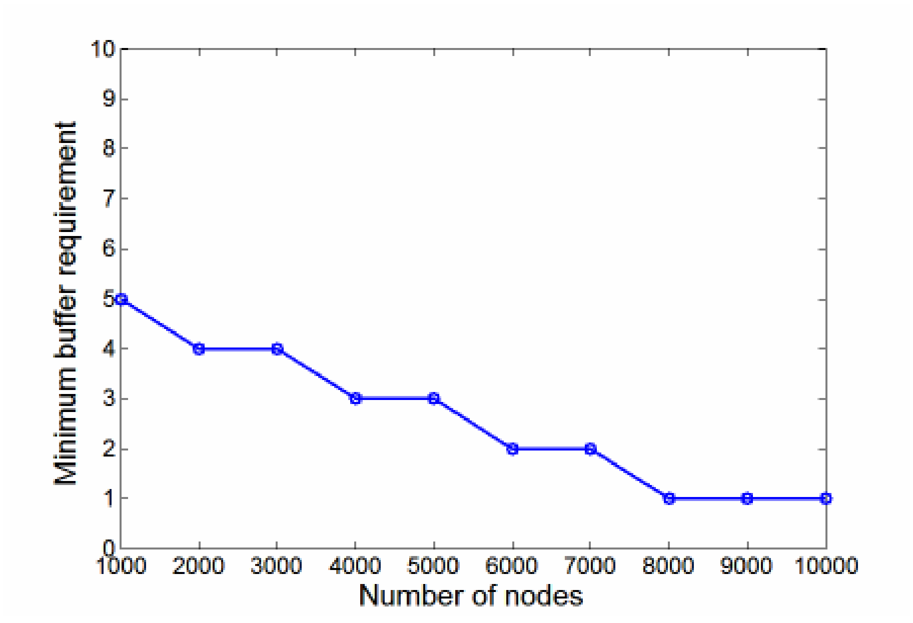


Figure 6.13 Minimum required buffer size as a function of group size

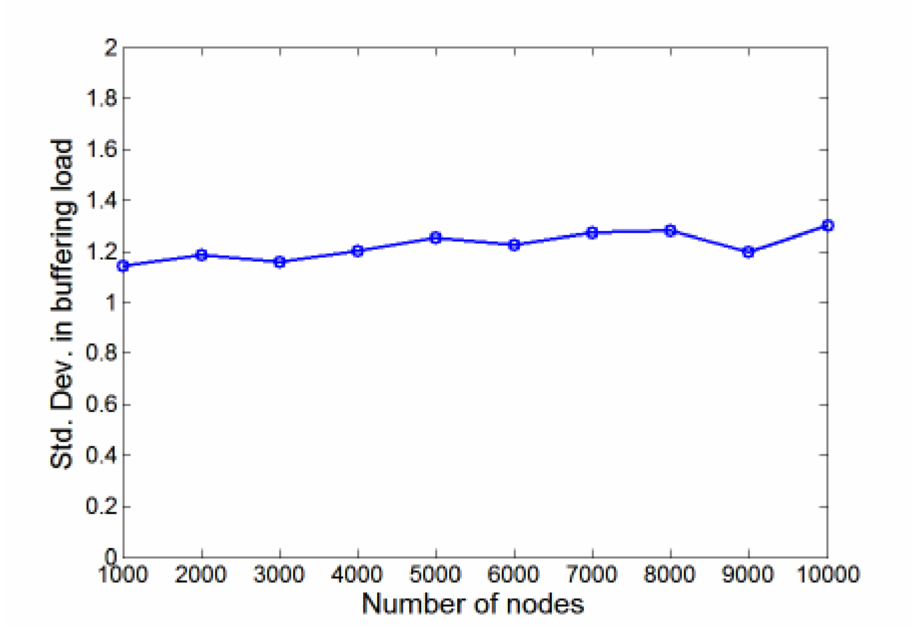


Figure 6.14 Standard deviation of buffering load as a function of group size

Fig. 6.15 shows the comparison of average message delays for Stepwise Fair-share Buffering with benchmark approaches as a function of network size. Stepwise Fair-share and Probabilistic buffering approaches result in higher average message delays per node, when compared to Hash-based and Random buffering, by the same reasoning given for Fig. 6.12 earlier. This is due to the fact that the former approaches use additional time to determine the bufferer of each data message disseminated. This result shows the drawback of not having the full membership information for the stepwise approaches. However, when distributing a large content consisting of thousands of messages, bufferer determination and message dissemination phases take place concurrently, and total dissemination time for the content is not affected adversely as depicted in Fig. 6.16. In this figure, content dissemination times are close to each other as the scale of the system increases. Hash-based method leads to a higher dissemination time since the bufferer peers receive the messages eventually during dissemination as explained in chapter 4.

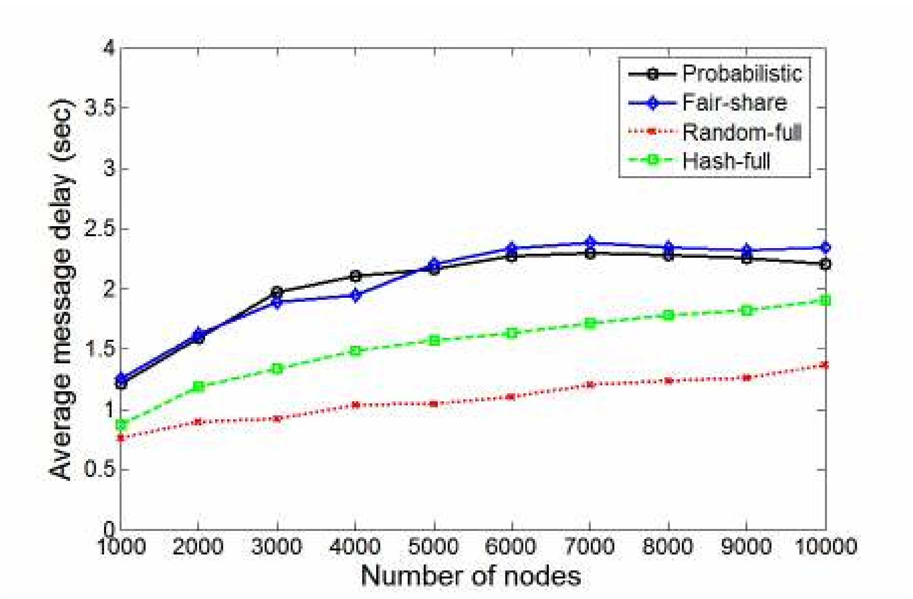


Figure 6.15 Comparison of average message delays as a function of group size

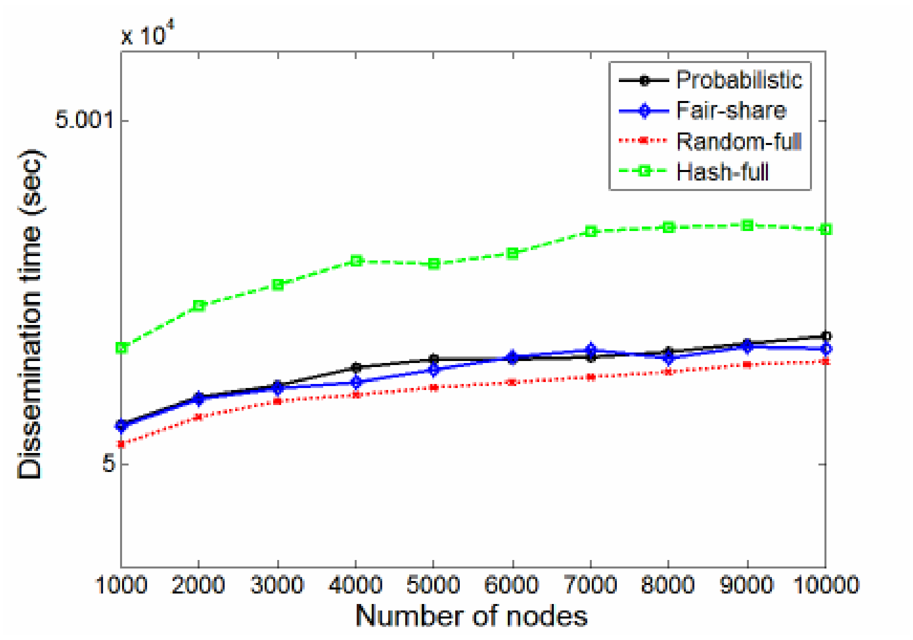


Figure 6.16 Comparison of dissemination time as a function of group size

6.4 Failure Cases

In this group of experiments, the performance of Stepwise Fair-share Buffering is investigated in case of link failures. For the experimental results provided up to this point, there is no failure in the links of the underlying network. Certain link drop probabilities are assigned for each message traversing the network. Suppose the link drop probability in the network is *p*, then any message traveling on the link has a chance of not being delivered to the destination with probability *p*. When the link drop probability is 0.01 any message reaches its destination node with 99 % chance in one link and with 95 % in one link if the probability is 0.05.

In the simulations given in Fig.s 6.17 and 6.18, a 1000 node network topology is used for message dissemination and 50000 messages are disseminated from a single source. Short-term buffer size of a peer is set to zero in order to observe the long-term buffer performance. The long-term buffer size is set to 5 because since minimum long-term buffer size is 5 if there is no link drop probability on the network as it can be observed from Fig. 6.12. The message generation rate is 100 messages /sec and the gossip interval is 200 msec. In the first result given in Fig. 6.17, the link drop probability of the network is increased from 0.01 to 0.05. Minimum number of bufferers needed for the reliability of the system is 5 messages. Fig. 6.18 gives the minimum buffer size needed for reliable dissemination when the number of bufferers *b* of a message is set to 6. Minimum buffer size is 5 messages if the drop probability is 0.01 as consistent with Fig. 6.17 and its settings. On the other hand, this increases to 11 if the drop probability is 0.05.

Fig. 6.19 and 6.20 show the comparative behavior of the systems as the drop probability of the links increases. In these simulations, 500000 messages are disseminated to the network, network size is 1000 peers, message generation rate is 100 message/sec, gossip interval is 200 msec. Short-term buffer size is 10, long-term buffer size is 20 and number of bufferers per message is 5 so that a reliable dissemination is achieved for all the scenarios. These simulations have the same settings with Fig. 6.19

and 6.20. Fig. 6.19 shows that average message delay increases as a function of the link failure rate. The average message delay in stepwise probabilistic and fair-share approaches is higher than the hash-based and random approaches throughout all failure rates. Content dissemination time also increases when the failure rate of the link increases. Hash- based approached leads to a greater dissemination time and Stepwise Fair-share Buffering shows a close performance to the other approaches with this metric.

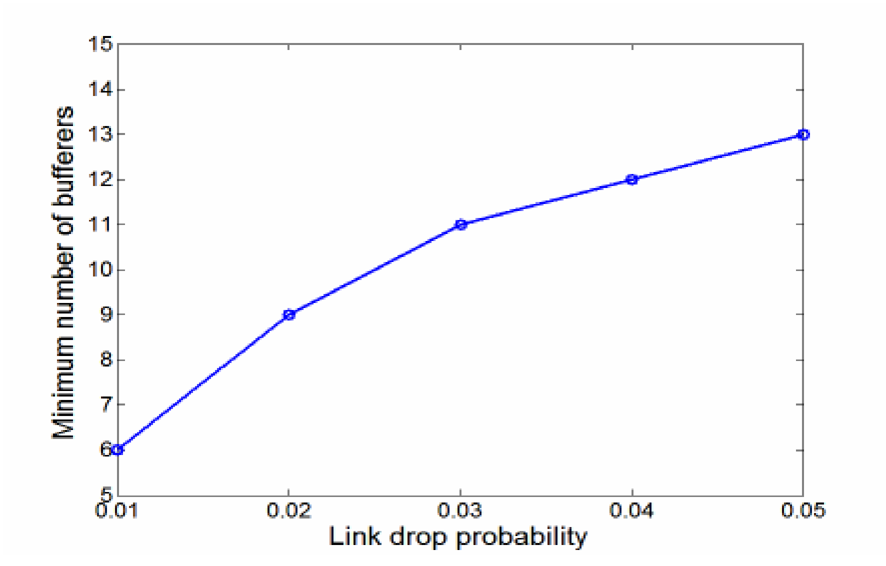


Figure 6.17 Minimum number of bufferers for reliability as a function of link drop

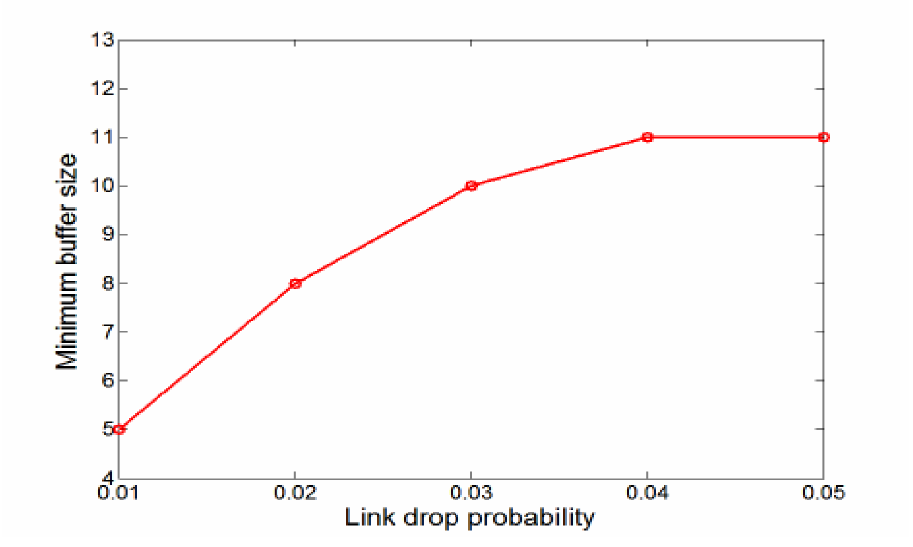


Figure 6.18 Minimum buffer size needed for reliability as a function of link drop probability

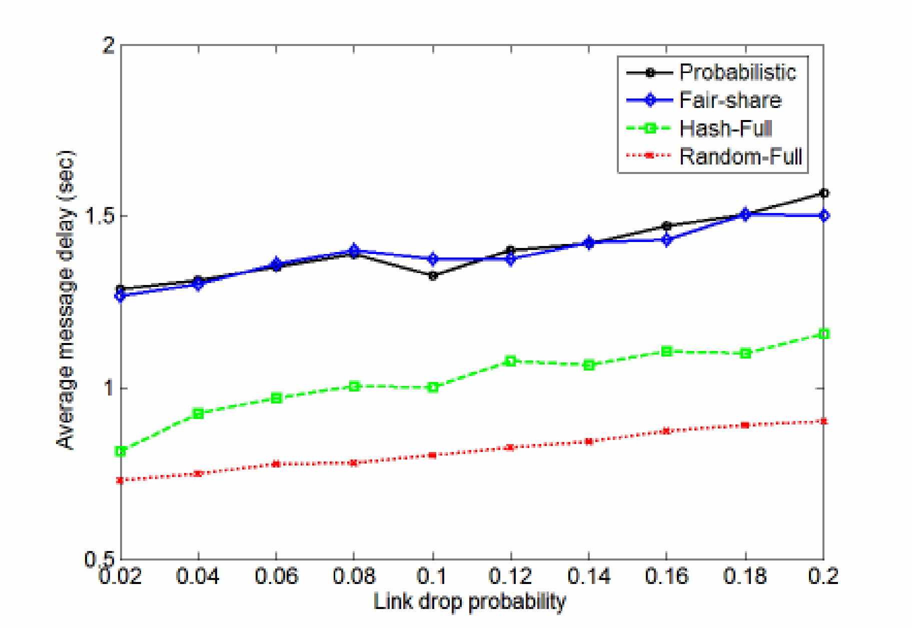


Figure 6.19 Comparison of average message delays as a function of link drop probability

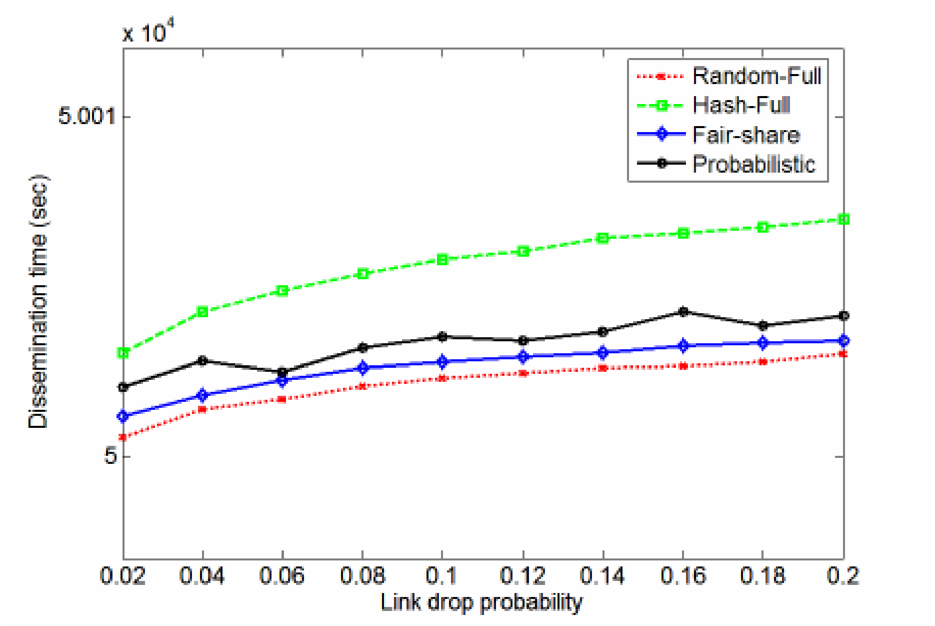


Figure 6.20 Comparison of dissemination time as a function of link drop probability

6.5 Distributing the Bufferers to Domains

As a last part of this chapter, distribution of bufferers to different domains is examined. In these simulations number of bufferers is set to 5, number of generated messages is 2000 and the 100 node topology is used. Fig. 6.21 gives the scattering ratio of the bufferers for Stepwise Fair-Share Buffering in the original case, namely when number of forwarders is equal to 1. Recall that scattering ratio is defined as the ratio of the number of distinct sub-domains a message is buffered to the total number of bufferers of the message. As it can be observed from Fig. 6.21 the bufferers of each message are distributed to one domain in the network. To increase the scattering ratio of the bufferers, number of forwarders parameter is introduced as in section 3.5 and is set to 15. This number is obtained by trial and error. Recall that the last forwarders is a list of nodes that the buffering request would not be sent. Also a modification on the algorithm is done for this purpose: when a buffering request is received, if the long-term buffer contains the corresponding message and STL is zero, then STL value is set to its initial value so that a different node can buffer the next copy of the same message.

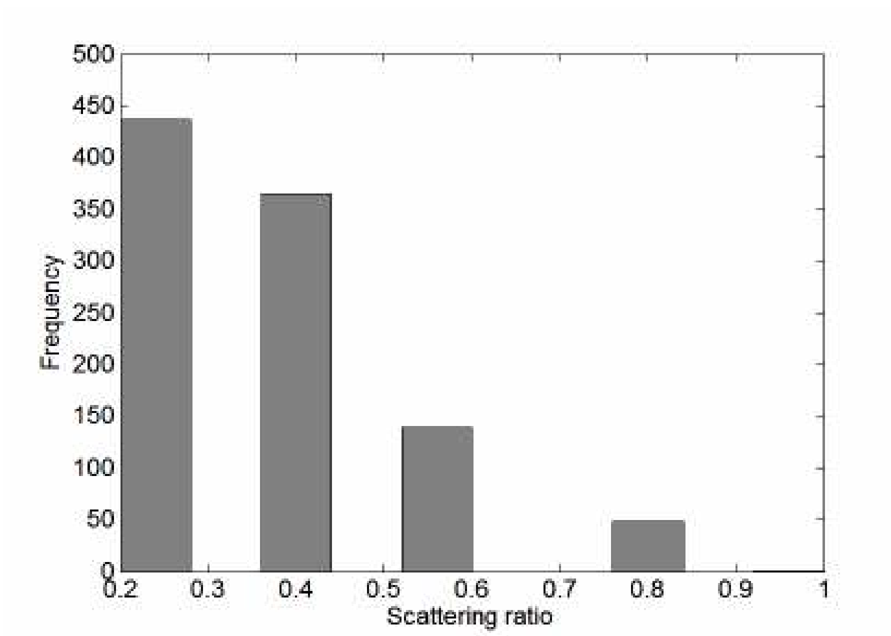


Figure 6.21 Distribution of bufferers to domains for number of forwarders=1

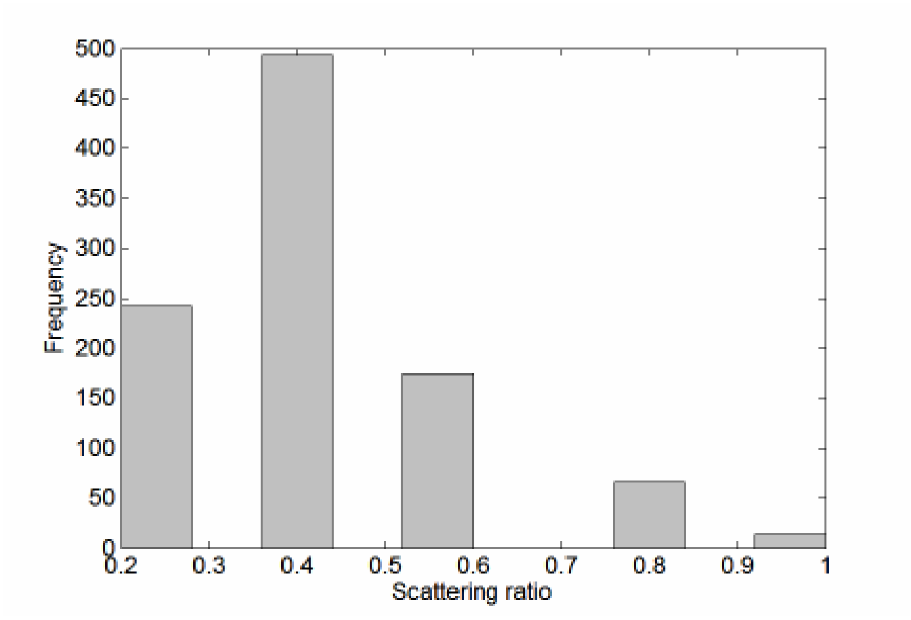


Figure 6.22 Distribution of bufferers to domains for number of forwarders=15

The scattering ratio for the modified version is given in Fig. 6.22. In this case scattering ratio increases in general. On the other hand, the uniformity of the scheme slightly decreases when compared to the original case as given in Fig 6.23.

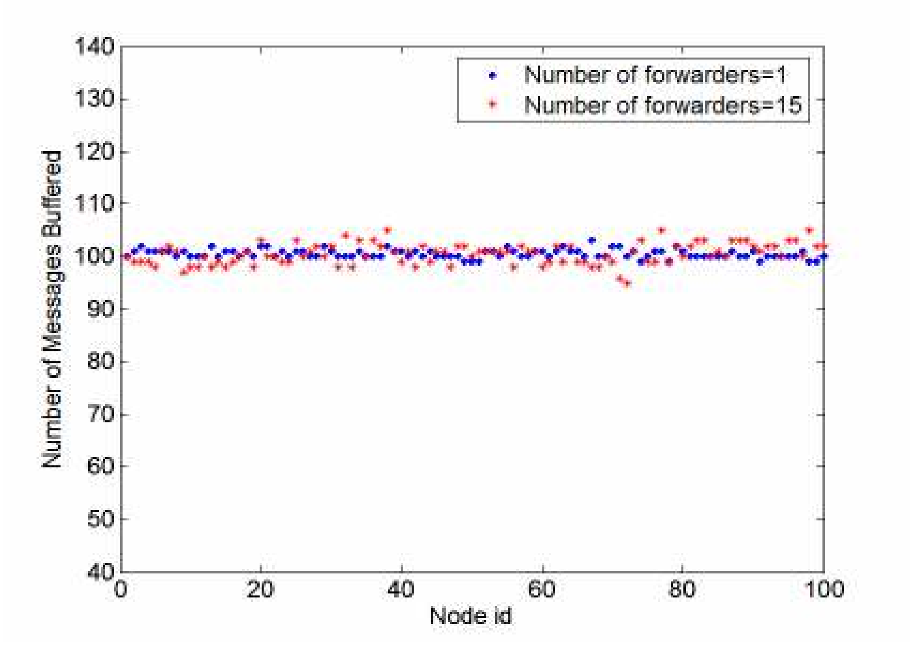


Figure 6.23 Comparison of buffering load with number of forwarders approach

Chapter 7

RELIABILITY BOUND FOR STEPWISE FAIR-SHARE BUFFERING

In this chapter, an analytical lower bound for reliability for each buffer size is obtained and compared with simulations. A Markov chain formulation is considered for finding the distribution of the dissemination time. Clearly, if the buffering time of a message is longer than its dissemination time, it can be safely discarded. In a random environment, the probability of this event represents the reliability of dissemination.

The main feature of Stepwise Fair-share Buffering is its long-term buffer mechanism and one of the major aims is providing reliability. In the experimental analysis of Chp. 6, the minimum long-term buffer requirement of the fair-share scheme is found for different networks and network conditions. In this section, a Markov Chain model is developed to find the buffer requirement of the peers. First mathematical preliminaries that are needed for the analysis are provided. Secondly, the details of the analytical model for the scheme are described. Then, a comparison of analytical work with the simulation results is given.

7.1 Markov Chains and Time to Absorption

A Markov chain $\{ X_n : n \geq 0\}$ is a stochastic process with a countable state space $J \subset Z_+$ that has the following property:

$$P[X_{n+1} = j \mid X_0 = i_0, \ldots, X_{n-1} = i_{n-1}, X_n = i] = P[X_{n+1} = j \mid X_n = i] = P_{ij} \qquad (1)$$

$$i_0, \ldots, i_{n-1}, i, j \in J \text{ and } n = 0, 1, \ldots$$

where the random variables X_n denote the states of the Markov Chain. The transition probabilities from one state to another are given by the transition probability matrix $P=[P_{ij}]$ which is a square matrix, $0 \leq P_{ij} \leq 1 \; i, j \in J$ and

$$\sum_{j=0}^{\infty} P_{ij} = 1 \quad i \in J \tag{2}$$

The states of a Markov Chain are classified according to the accessibility from one to another. A state i is an *absorbing state* if $P_{ii}=1$. Such states do not communicate with the other ones namely if the process enters an absorbing state it cannot leave the state. A state i is a *transient state* if $P[X_n = i, X_{n-1} \neq i, ..., X_1 \neq i \mid X_0 = i] < 1$, namely there is a non-zero probability that the process will never visit state i. A state is *recurrent* if $P[X_n = i, X_{n-1} \neq i, ..., X_1 \neq i \mid X_0 = i] = 1$. This means that the state will be revisited again in the future.

The transition matrix P of an absorbing Markov chain with t transient states and r absorbing states can be written in the form of a fundamental matrix such that

$$P = \begin{bmatrix} Q & R \\ 0 & I \end{bmatrix} \tag{3}$$

where Q is a t by t matrix that consists of the transient states, R is a r by r matrix that consists of the absorbing states. 0 is the zero matrix that is t by r and I is the r by r identity matrix.

The probability mass function f, the cumulative distribution function F and the expected value μ of the time to absorption of a Markov chain containing absorbing states can be found explicitly as

$$f(k) = \begin{cases} \beta_0 & k = 0 \\ \beta Q^{k-1} R & k \geq 1 \end{cases} \tag{4}$$

$$F(k) = \beta_0 + 1 - \beta Q^k e \quad k \geq 0 \tag{5}$$

$$\mu = \beta (I - Q)^{-1} e \tag{6}$$

where $k \in Z_+$ denotes the time to absorption, β_0 and β denote the probability and the probability vector that the Markov Chain starts at the absorbing and transient states, respectively, and e is the vector consisting of all 1's [35,36].

7.2 Reliability Bound for Stepwise Fair-share Buffering

In this section, the details of the analytical model that have been developed for the buffering scheme are explained. Let α denote the message generation rate of the source node, λ denote the rate of receiving a new message to be buffered, n denote the number of nodes in the system, B denote the size of the long-term buffer of a node (namely number of messages in the long-term buffer when the buffer is full) and T denote the time that passes for one message to reach all the nodes. Our aim is to find the minimum buffer size B that guarantees reliable delivery of a message to all nodes.

Let the rate of receiving a new message to buffer be denoted by λ for a given node. Then, we can approximate the expected time between two buffer updates by $\frac{1}{\lambda}$. Since FIFO replacement scheme is used in Stepwise Fair-share Buffering, a message that has been recently received will be dropped from the buffer if B new messages are received after the reception of that message as illustrated in Fig. 7.1.

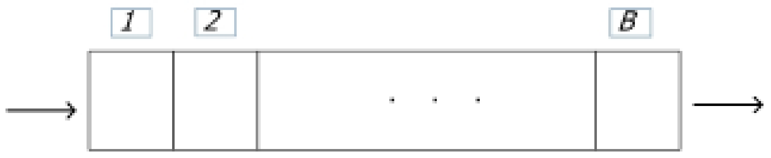


Figure 7.1 FIFO replacement scheme

So, the expected waiting time of one message in the long-term buffer of a node can be approximated as:

$$W = \frac{1}{\lambda} \cdot B \tag{7}$$

By virtue of the results obtained in the experiments, it can be assumed that load of being bufferer is distributed uniformly to all n nodes. Thus, the rate of being bufferer can be computed as:

$$\lambda = \frac{\alpha}{n} \tag{8}$$

Therefore, in view of (7) and (8) the average waiting time of one message in the long-term buffer of a node becomes

$$W = \frac{n}{\alpha} \cdot B \tag{9}$$

To provide perfectly reliable dissemination, the waiting time of one message in the long-term buffer of a node should be greater than the time T that passes for one message to reach all the nodes. That is, we must have $W \geq T$ in order to have a reliable dissemination. Approximating the waiting time in the buffer as a deterministic quantity with its average value (9), we require $F\left(\frac{Bn}{\alpha}\right) = P[T < \frac{Bn}{\alpha}]$ to 1 or very close to 1. Clearly this serves as a lower bound for reliability in presence of short-term buffers. Hence, the missing part is the computation of F. For this purpose, a Markov Chain model is developed for epidemic dissemination of messages. Suppose that the states of the Markov chain $\{X_t : 0 \leq X_t \leq n\ ,\ t = 0,1,2,\ldots\ \}$ are defined as the number of infected nodes for one message in the system at time t. The transition probabilities from one state to another will be determined by the epidemic mechanism. This is an absorbing Markov chain and absorbing state is the total number of nodes n. Therefore, the results in (4) , (5) and (6) will be used to find the distribution of T.

Exact transition probabilities for different epidemic dissemination models are obtained in [37]. The models are pull, push and hybrid. In the pull model, an infectious peer selects a susceptible peer randomly and sends its digest message to a susceptible peer. In the push model, the process is the reverse namely a susceptible peer selects an infectious peer randomly and sends its digest message to the infectious peer. The hybrid model is the combination of these two models. In the analytical analysis below, push model is used. In the push model, the transition probability that there will be j infected nodes at the next stage when there are k infectious peers at present is found as:

$$P_{kj} = P[I_{t+1} = j \mid I_t = k] = \frac{\binom{n-k}{j-k} k^{j-k} (n-k-1)^{n-j}}{(n-1)^{n-k}}, j=1,2,\ldots,n-k \tag{10}$$

Using this information the matrices Q and R are constructed and used in Equation (5). The cdf $F(k) = \beta_0 + 1 - \alpha Q^k e$, is evaluated for $k = \left\lfloor \frac{Bn}{\alpha} \right\rfloor$ because k should be an integer. In this model β which is the probability vector that the Markov chain starts at the transient state is (1,0,…,0) since initially only the source node has the copy of the message and β_0 which is the probability that the Markov chain starts at the absorbing state is 0. Then, the following result is obtained:

$$F\left(\frac{Bn}{\alpha}\right) = 1 - [1 \quad 0 \quad \ldots \quad 0] Q^{\left\lfloor \frac{Bn}{\alpha} \right\rfloor} [1 \quad \ldots \quad 1]^T = 1 - \sum_{j=1}^{n-1} Q_{1j}^{\left\lfloor \frac{Bn}{\alpha} \right\rfloor} \tag{11}$$

Therefore, $F\left(\frac{Bn}{\alpha}\right)$ is equal to 1 - (sum of the first row of $Q^{\left\lfloor \frac{Bn}{\alpha} \right\rfloor}$) . Using this information the minimum buffer size B needed for reliable dissemination is computed for each level of reliability.

The transition probability that is given in (10) for push model uses the assumption that the fan-out parameter is 1. For fan-out parameter f greater than 1, the transition probability that is computed in [38] is used. It is given by

$$P_{kj} = \binom{n-k}{j-k} \left(1 - \frac{\binom{n-k}{f}}{\binom{n}{f}} \right)^{j-k} \cdot \left(\frac{\binom{n-k}{f}}{\binom{n}{f}} \right)^{n-j}, \ j=1,2,\ldots,n-k \tag{12}$$

with the slight correction that the susceptible peer can select peers among n-1 possibilities excluding itself.

7.3 Comparison with Simulation Results

In this section, the results obtained from the analytical model explained above are compared with the results obtained from the simulation of Stepwise Fair-share Buffering scheme. Reliability of the scheme with different buffer sizes in analytical and simulation results are compared. In the evaluations, there are 100 node in the system ($n = 100$), message generation rate α is 100 messages/sec. 100 simulations are performed for each point on the graphs and their average is taken.

In Fig. 7.2, analytical and simulation results are compared when fan-out is 1. In other words, for the matrix Q the probabilities given in (10) are used. As it can be inferred from the figure, the results obtained from the analytical computations and simulations are close to each other for higher buffer sizes and different for the smaller buffer sizes. The discrepancy occurs in small buffer sizes because, in the push model the nodes have full membership information and in the simulations the nodes have partial membership information. Besides, the waiting time $\frac{Bn}{\alpha}$ is only an average value. On the other hand, the analytical results are consistent with the simulation results for large reliability values. Therefore, the analytical model can be used for designing a highly reliable system.

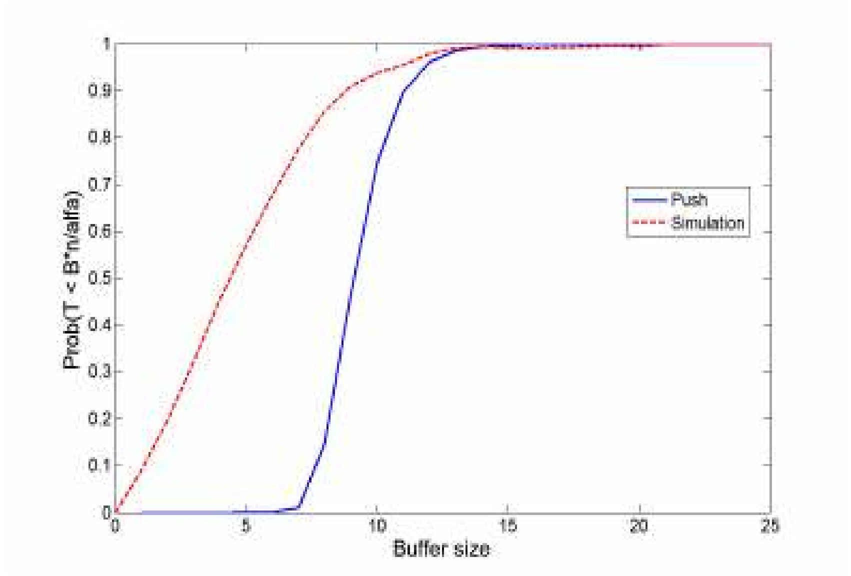


Figure 7.2 Reliability versus buffer size for model and simulation (f=1)

The results are similar to *f*=1 case except that the reliability is achieved with a smaller buffer size since the increase in fan-out increases the speed of the epidemic spread. So, a message reaches all nodes in a shorter time period and smaller buffer size becomes enough in this case.

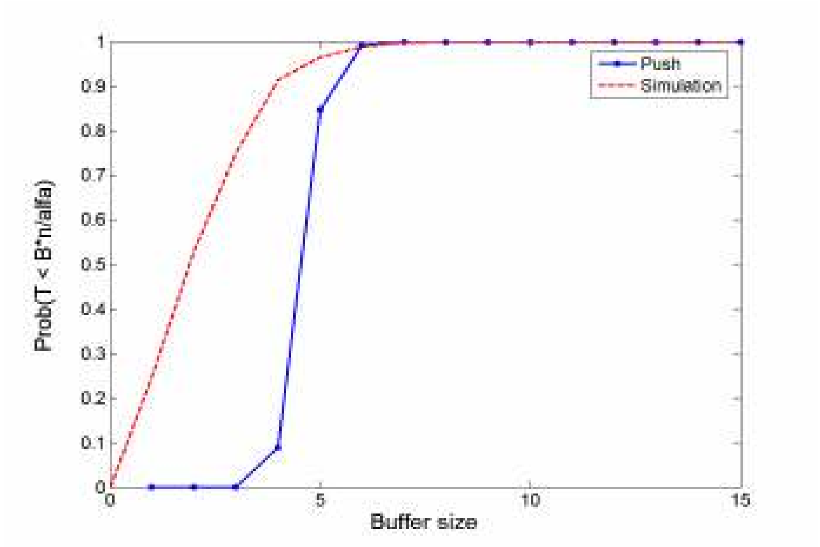


Figure 7.3 Reliability versus buffer size for model and simulation (f=3)

In Fig. 7.4, the reliability computed by the analytical model is compared for different fan-out parameters. As expected, if fan-out increases, the same buffer size provides more reliable dissemination.

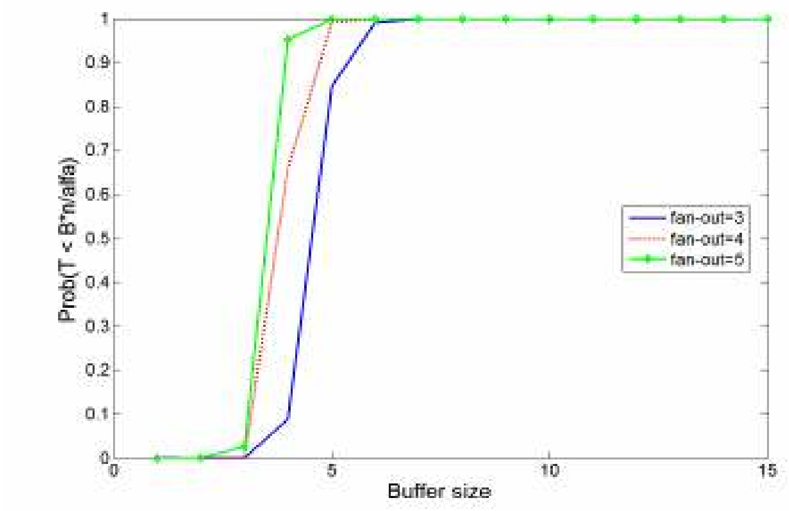


Figure 7.4 Comparison of the analytical results for different fan-out values

Chapter 8

CONCLUSION AND FUTURE WORK

In this thesis, buffer management problem for P2P epidemic information dissemination systems is investigated. Novel buffer management models are developed for these systems. Performance evaluation of these models is done using simulation and analytical models.

The first model used is the Stepwise Probabilistic Buffering that distributes the load of buffering to the entire system where all peers have partial knowledge of the overlay. It reduces the memory usage; it is applicable to dissemination of data to a large group of peers where epidemic dissemination idea is used. In this study, the existence of an overlay among peers reflecting the properties of the underlying network topology is assumed, and transit-stub model is used which is a good approximation of the Internet topology. It is shown that Stepwise Probabilistic Buffering scheme distributes the buffering load to all peers, reduces the dissemination time and the buffer space of all data, and improves the utilization of buffers and the reliability of dissemination.

A more robust scheme Stepwise Fair-share Buffering is developed to remove the inconvenience of the probabilistic algorithm for a large scale network where the computations of the probabilities are infeasible. Stepwise Fair-share Buffering provides a more uniform buffering load distribution among the peers. It is scalable, simple and applicable to any kind of underlying network topology. It does not impose additional overhead on the buffering request message still with only partial knowledge of the system. As a result, the efficiency of content dissemination is improved.

The performance of the buffering approaches has been evaluated through simulations. Hash-based buffering scheme and a completely randomized approach with full membership information are used as benchmark for comparison. Stepwise Fair-share Buffering performs well; it is scalable for large networks also in the case of any failures in the links. Thus, it can take place of the Hash-based buffering scheme.

Analytical results for reliability of epidemic dissemination as a function of buffer levels are derived. These results are based on a Markov chain analysis and are evaluated numerically. Comparison with simulations of Stepwise Fair-share scheme shows that the analytical model provides a good lower bound for reliability. For high level of reliability values, the bounds are very close to the simulation results

As future work, we aim to include link failures in the underlying network topology to the analytical model. In case of link failures, one bufferer will not be sufficient for reliable delivery of a message. We plan to compute the minimum number of bufferers required for reliability in this scenario and compare the analytical results with the simulations.

In order to measure the accuracy of our simulator, an application can be developed and deployed on a set of testbed nodes. Then, the simulation results for the buffering algorithms can be compared with those obtained from the testbed.

BIBLIOGRAPHY

[1] R. van Renesse, Y. Minsky, and M. Hayden, “A Gossip-Style Failure Detection Service”, Int’l Conf. Distributed Systems and Platforms and Open Distributed Processing (IFIP’98), 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] 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, 1999, pp. 188-203.

[5] 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. USA, 2002.

[6] J.C. Lin and S. Paul, “RMTP: A Reliable Multicast Transport Protocol,” Proc. of the 15th IEEE Conf. on Computer Comm. (INFOCOM’96), San Francisco, USA, 1996, pp. 1414 - 1424.

[7] S. Floyd, V. Jacobsen, C.G. Liu, S. McCanne, and L. Zhang, “A Reliable Multicast Framework for Lightweight Sessions and Application-Level Framing,” IEEE/ACM Trans. on Networking, vol. 5, no. 6, Dec. 1997, pp. 784-803.

[8] S. Mishra and L.Wu, “An evaluation of flow control in group communication”, IEEE/ACM Trans. on Networking, vol. 6, no. 5, Oct. 1998, pp. 571-587.

[9] K. 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. 2313-2316.

[10] 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'03), San Francisco, CA, USA, 2003.

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

[12] K. Guo and I. Rhee, "Message Stability Detection for Reliable Multicast," Proc. of the 19th IEEE Conf. on Computer Comm. (INFOCOM'00), New York, USA, 2000, pp. 814-823.

[13] 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, 1999, pp. 1256-1264.

[14] J. Pereira, L. Rodrigues, M. Monteiro, R. Oliviera, A. M. Kermarrec, "Network Friendly Epidemic Multicast", 22nd International Symposium on Reliable Distributed Systems (SRDS'03), Florence, Italy, 2003.

[15] C. Lindemann and O. Waldhorst, "Modeling Epidemic Information Dissemination on Mobile Devices with Finite Buffers", Proc. of the ACM. Int. Conf. on Measurement & Modeling of Computer Systems (SIGMETRICS'05), Banff, Canada, 2005, pp. 121-132.

[16] A. Dan and D. Towsley, "An Approximate Analysis of FIFO and LRU Buffer Replacement Schemes", Sigmetrics'90, ACM Press, 1990.

[17] E. Ahi, M. Cağlar, Ö. Özkasap, "Stepwise Probabilistic Buffering for Epidemic Information Dissemination", Bio-inspired Models of Network, Information and Computing Systems (Bionetics'06), Cavalese, Italy, 2006.

[18] E. Ahi, M. Çağlar, Ö. Özkasap, “Stepwise Fair-Share Buffering underneath Bio-inspired P2P Data Dissemination”, International Symposium on Parallel and Distributed Computing (ISPDC’07), Hagenberg, Austria, 2007.

[19] K.P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, “Bimodal Multicast,” ACM Trans. on Computer Systems, vol. 17, no. 2, May 1999, pp. 41-88.

[20] D. D. Clark and D. L. Tennenhouse, “Architectural considerations for a new generation of protocols”, Proc. of ACM SIGCOMM, 1990.

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

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

[23] A. Rowstron, P. Druschel, “Pastry: Scalable, distributed object location and routing for large scale peer-to-peer systems”, Proc. of the IFIP/ACM Middleware, Heidelberg, Germany, 2001.

[24] 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.

[25] B. Koldehofe, “Buffer Management in Probabilistic Peer to Peer Communication” Proceedings of the 22nd International Symposium on Reliable Distributed Systems (SRDS’03), IEEE, Florence, Italy, 2003.

[26] R. Motwani and P. Raghavan, “Randomized Algorithms”, Cambridge University Press, Cambridge, England, June 1995.

[27] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, “Performance Modeling of Epidemic Routing” University of Massachusetts Technical Report CMPSCI 05-44, 2005.

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

[29] M. Jelasity, A. Montresor, and G. P. Jesi. “Peersim Peer-to-Peer Simulator”, 2004. http://peersim.sourceforge.net/.

[30] K. Kant and R. Iyer, “Modeling and Simulation of Adhoc/P2P Resource Sharing Networks”, Proc. of the TOOLS, Illinois, USA, 2003.

[31] http://pdos.csail.mit.edu/p2psim

[32] E. Ahi, M. Çağlar, Ö. Özkasap, “Message Buffering in Epidemic Data Dissemination”, International Symposium on Computer Networks (IEEE ISCN’06), İstanbul, Turkey, 2006.

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

[34] E. W. Dijkstra: “A note on two problems in connexion with graphs.”, In: Numerische Mathematik. 1, 1959, pp. 269–271

[35] T.Issariyakul, E. Hossain, and A. S. Alfa “Analysis of Latency for Reliable End-to-End BatchTransmission in Multi-Rate Multi-Hop Wireless Networks”, Proc. Of the IEEE ICC’05, Seoul, Korea, 2005.

[36] M. F. Neuts, “Matrix-geometric solutions in stochastic models”, The John Hopkins University Press, 1981.

[37] O. Özkasap, E. Yazıcı, S. Küçükçifçi, M. Çağlar, “Exact Performance Measures for Peer-to-Peer Epidemic Information Diffusion” International Symposium on Computer and Information Sciences (ISCIS’06), İstanbul, Turkey, 2006.

[38] M. Çağlar, Ö. Özkasap, “A Chain-Binomial Model for Pull and Push-Based Information Diffusion”, International Conference on Communications (ICC’06), Istanbul, Turkey, 2006.

VITA

Emrah Ahi was born in Ankara, Turkey on April 9, 1981. He received his B.Sc. degree in Mathematics from Middle East Technical University, Ankara, in 2004. From September 2004 to August 2006, he worked as a teaching and research assistant in Koç University, Istanbul, Turkey and participated in the “TUBITAK-COST Action: 279, Analysis and Design of Multiservice Networks Supporting Mobility, Multimedia, Internetworking” project. He is supported by TUBITAK CAREER Project 104E064. He presented a technical report at COST-279 ’05, (Antalya, Turkey) and he has four conference papers presented at PODC ’05 (Las Vegas), ISCN (Istanbul, Turkey), FAE ’06 (Lefke, Cyprus) and Bionetics ’06 (Cavalese, Italy). Additionally he will present a conference paper in ISPDC ’07 (Hagenberg, Austria). He is currently working as a software engineer in Risk Software Technologies, İstanbul, Turkey.