NEEM: Network-friendly Epidemic Multicast[*]

J. Pereira
U. do Minho
jop@di.uminho.pt

L. Rodrigues
U. de Lisboa
ler@di.fc.ul.pt

M. J. Monteiro
U. de Lisboa
mjmonteiro@di.fc.ul.pt

R. Oliveira
U. do Minho
rco@di.uminho.pt

A.-M. Kermarrec
Microsoft Research
annemk@microsoft.com

July 28, 2003

Abstract

Epidemic, or probabilistic, multicast protocols have emerged as a viable mechanism to circumvent the scalability problems of reliable multicast protocols. However, most existing epidemic approaches use connectionless transport protocols to exchange messages and rely on the intrinsic robustness of the epidemic dissemination to mask network omissions. Unfortunately, such an approach is not network-friendly, since the epidemic protocol makes no effort to reduce the load imposed on the network when the system is congested.

In this paper, we propose a novel epidemic protocol whose main characteristic is to be network-friendly. This property is achieved by relying on connection-oriented transport connections, such as TCP/IP, to support the communication among peers. Since during congestion messages accumulate in the border of the network, the protocol uses an innovative buffer management scheme, that combines different selection techniques to discard messages upon overflow. This technique improves the quality of the information delivered to the application during periods of network congestion. The protocol has been implemented and the benefits of the approach are illustrated using a combination of experimental and simulation results.

1 Introduction

Reliable multicast is a fundamental building block of many distributed applications. However, providing stable high throughput to large groups while, at the same time, enforcing strong reliability is a very hard task [1]. Some protocols, such as RMTP [15], generate a large number of acknowledgments that must be received and processed, a task that may rapidly overwhelm the sender. Even with scalable acknowledgment mechanisms, messages have to be buffered and retransmitted until all recipients acknowledge their reception or are declared failed [9]. Therefore, when a receiver is

[*]Selected section of this report have been published in the Proceedings of the 22nd Symposium on Reliable Distributed Systems, October, 2003, Florence, Italy. Work partially supported by FCT project RUMOR (POSI/40088/CHS/2001) and Microsoft Research Grant (2001-39).

perturbed, messages accumulate and eventually prevent further messages to be sent for some time. This means that a single slow receiver slows down the entire group [24].

Epidemic multicast protocols are an attractive proposal to address the performance and scalability issues of reliable multicast. These protocols, also called gossip-based or probabilistic, support the efficient dissemination of data among a large number of nodes while providing a probabilistic guarantee of delivery [11, 2, 13, 4]. Epidemic multicast protocols support throughput stability even for very large groups with perturbed nodes, as the load required to ensure reliability is evenly spread across all members of the group and no single perturbed node can block senders. Additionally, gossiping offers high resilience to node and network failures.

The probabilistic atomic delivery guarantee of epidemic protocols is based on the assumption of an uniformly distributed message loss. This assumption does not hold if the input load exceeds the capacity of the network. When the network is congested, sequences of messages can be discarded, leading to failure of the protocol. This phenomenon is exacerbated by the usage of connection-less transport layer such as UDP/IP for exchanging gossip messages. As UDP/IP based protocols do not usually perform congestion control, routers are often configured to discard UDP/IP traffic when congested. This is a sensible choice, as failing to perform congestion control can lead to a severe degradation of throughput in congested networks [12, 5]. Additionally, in todays Internet, participants in multicast traffic may access the network through residential connections such as ADSL, cable, or even traditional analog modems. Even broadband connections are asymmetric and have restricted uplink capacities. This further complicates the deployment of epidemic protocols which generate symmetric traffic.

In this paper, we propose a novel epidemic protocol called NEEM, Network-friendly Epidemic Multicast. As its name implies, the main goal of the protocol is to ensure that the protocol does not negatively contribute to the congestion of the network during overload periods. In order to do so, our proposal combines a standard epidemic protocol with the following complementary mechanisms:

- TCP-friendly [5] end-to-end congestion control, allowing safe usage of available bandwidth. This property is achieved by relying on connection-oriented transport connections, such as TCP/IP, to support the communication among peers.
- An innovative buffer management technique at the border of the network, that combines different selection techniques to discard messages on overflow. Using this approach, it is possible to preserve the throughput stability of epidemic protocols while at the same time improve the quality of the information delivered to the application during network congestion periods.

A key feature of our buffer management technique is the usage of message semantics to select which messages to discard upon overflow. This work shows that a technique that has been used to improve throughput stability of deterministic reliable multicast protocols [23] can also benefit epidemic dissemination. The approach stems from the observation that in distributed applications messages often overwrite the content of others sent shortly before, therefore making them obsolete while still in transit. We have previously suggested the usage of message semantics in probabilistic multicast protocols in a short paper [21]. The current paper includes the following novel significant contributions: *i)* we describe in detail the architecture for combining probabilistic multicast and semantic reliability and introduce its implementation using TCP/IP; *ii)* we use a simulation model to illustrate the performance of the proposed architecture, showing how configuration parameters and design decisions, such as buffer size and purging policies, affect the overall performance of the protocol; and *iii)* we present results with traffic from a real application.

The paper is organized as follows. Section 2 motivates our work by describing the target application and network environment, introducing also the intuition behind our approach. The proposed protocol is described in Section 3. Section 4 evaluates the proposed protocol and identifies the configuration parameters that offer the best results. Finally, Section 5 discusses related work and Section 6 concludes the paper.

2 Motivation

2.1 Target Applications

Most applications that need to periodically disseminate updates to a large number of recipients may benefit from a protocol such as NEEM. On one hand, the scalability requirements invalidate the use of conventional reliable multicast protocols. Epidemic protocols offer a good balance among reliability and scalability. On the other hand, the periodic nature of traffic makes it highly likely that the message flow exhibits some level of redundancy. Such redundancy can be exploited by a clever buffer management scheme to selectively discard messages during overflow periods. Examples of applications with these characteristics are multi-player games, and real-time multicasting of data from sport events to a very large number of observers, such as tracking of GPS coordinates of participants in the Paris-Dakar or the American Cup. Other examples of traffic with similar characteristics can be found in [22].

Although the applicability of our proposal is not limited to a single application, to make our discussion concrete we focus on multi-player games. The growing popularity of multi-player networked games has sparked an interest in supporting large number of participants over the Internet, both as players and as observers. Player ranks are organized and matches between top players attract considerable attention. It has even been suggested that on-line games are turning into a spectator sport, in which the number of spectators tends to be far greater than the number of participants [3]. In fact, observers are expected to be a source of revenue for hosting companies. Namely, it is attractive to use on-line games as an advertisement medium. Providing the observer capability is also regarded as a mean to increase gaming hardware and software sales.

As hosting games becomes commercially interesting, there is a demand for specialized middleware which eases their development and deployment. Current multi-player games and existing toolkits are either based on a centralized server, which maintains the state of the game, or on a peer-to-peer model in which each player disseminates some of the information to all other participants. Neither of these is suitable for coping with hundreds of spectators. In addition, it is expected that spectators are not passive. Support for complementary activities, already common in current spectator sports, such as chatting and betting[1] can significantly increase the quality of the experience.

2.2 Case Study: Microsoft Flight Simulator 2002

Microsoft Flight Simulator 2002 (FS2002) is a realistic flight simulation game which allows for multi-player operation on local area networks and on the Internet. Each player controls a single aircraft in a common virtual world, where they can interact visually, by colliding, and by exchanging text messages. The implementation of

[1] As an example visit `http://youplaygames.com/`.

FS2002 uses the DirectPlay peer-to-peer API on the Windows operating system [17]. This toolkit provides a simple membership and multisend protocol. The state of the game is replicated by all participants. Each player maintains the authoritative state of the locally controlled aircraft and periodically updates other participants by sending information about its position and velocity [18]. Other messages are used for additional gaming facilities (e.g. chat messages, the external look of the plane, name of the player) and for membership management.

Let us analyze the communication requirements of a game such as FS2002. Each player multicasts 4 state update messages each second, carrying 60 bytes of payload each. When players are connected using V.90 modems (56 kbps downlink/33.6 kbits uplink bandwidth) this translates to a maximum of 17 destinations of each state update. As each player is responsible for multicasting its updates to all observers, this means that 17 is in fact the total number of players plus spectators. Even with broadband connections, the uplink is lower than the available downlink bandwidth, thus limiting the number of destinations. This can be mitigated by relying on a centralized server with a more expensive high bandwidth network connection, as can be provided by a commercial game hosting service. This does not however entirely solve the problem, as a hosting service is also expected to host more than one simultaneous instance of the game. The amount of traffic imposed on the server is still proportional to the total number of spectators and thus the operating cost of such service grows linearly with the bandwidth used. Ideally, one would take advantage of the observers themselves to relay the traffic, thus removing the dependence on the servers and the upper limit on the number of spectators.

2.3 Epidemic Multicast on the Internet

At first sight, it seems that the use of epidemic multicast protocols [11, 2, 13, 4] would trivially make it possible to support a very large number of observers in a multi-player game without imposing a proportional load on game servers. Unfortunately, the deployment of an epidemic protocol on the envisaged network environment is problematic. As noted before, since the probabilistic guarantees are based on the assumption of packet loss being uniformly distributed across the network and in time, epidemic protocols do not cope well with congestion that causes bursts of lost packets in a single link. In fact, the high probability of message loss during congestion periods is not only due to the congestion itself, but also due to the fact that epidemic protocols are typically based on network-unfriendly connectionless transport protocols, such as UDP/IP [12]. Internet service providers (ISPs) often configure their networks to prioritize TCP/IP traffic to ensure stable performance. Thus, when a network link is being fully used, the UDP/IP traffic is therefore more likely to be discarded.

Given that several low bandwidth links are expected to exist in our target network environment, and that these links may be shared by different applications, congestion periods are bound to happen. Therefore, current epidemic protocols are not appropriate to disseminate updates from highly dynamic multi-user games to a large set of nodes. To address these problems, we propose a new epidemic multicast protocol that is based on network-friendly, connection-oriented transport protocols, such as TCP/IP, to support the communication among peers. A consequence of such approach is that, during congestion periods, messages are not injected on the network but, on the contrary, accumulate at the network border, under control of the epidemic protocol. This raises the opportunity to be selective when deciding which messages to discard when buffers overflow. Our selection procedure uses a combination of criteria, among which is the use of semantic knowledge about the message contents, as discussed below.

2.4 On the Use of Semantic Knowledge for Buffer Management

Consider the traffic generated by Flight Simulator 2002. The bulk of the traffic are periodic messages that convey each airplane's position to other players and to observers. In general, each update makes the preceding information about the same airplane obsolete, as the most current information is always preferable. Given the nature of the epidemic dissemination, messages are retransmitted a number of times. Each message stays within the system for some time. It is thus likely that some messages are already obsolete while still being relayed. Notice that messages are usually delivered to the vast majority of destinations in less hops than those necessary to ensure probabilistic atomicity. This means that messages that are already obsolete and that have even been delivered can still consume network bandwidth. If such messages could be recognized by intermediate nodes, a significant amount of bandwidth could be saved.

From the previous paragraph, it may seem that the naive solution of discarding all but the latest message sent by each source is a effective selection criteria. To achieve good results, one needs a more sophisticated approach. First, some of the messages are not related to the airplane position and thus should be delivered with the highest reliability possible. Second, not all position updates become obsolete. Those updates that lead to the discovery of a collision or the crossing of a finish line should be reliably delivered to all players. Observers may also value the ability to reconstruct the entire route and thus updates that correspond to sharp changes of direction should not be discarded. Finally, in applications other than FS2002, a single node might manage more than one object, thus multicasting unrelated streams of update messages that should be managed separately. It is therefore required that some amount of message semantics is known by the protocol. Our goal is thus to build a general purpose protocol that simultaneously takes advantage of message semantics, to reduce bandwidth requirements, but that is not tied to the content of messages produced by a single application.

3 The NEEM Protocol

Our novel Network-friendly Epidemic Multicast protocol (NEEM) is obtained by combining three complementary layers. A simple epidemic protocol [11] embodying partial membership management [4] is the top layer. This layer is briefly described for completeness, as it has been introduced and is thoroughly described elsewhere. The next two sections introduce the specialized buffer management layer and the configuration of the transport layer, which are the core of our proposal. We then conclude by discussing how the three layers interact to achieve the desired goal.

3.1 Epidemic Dissemination

Epidemic multicast works as follows: A message is initiallu tagged with the maximum number of rounds r and then forwarded to distinct f other nodes chosen randomly. On reception, the number of rounds remaining is decremented. If zero remain, the message is discarded. If not, the message is forwarded again to another f nodes. Delivery happens when a new message is received by a node. The guarantees offered by the protocol depend on appropriate configuration of r and f [11, 13]. The membership protocol is itself based on gossip and keeps at each node list of other locally known nodes [4]. This list is a partial view of the entire group, as it has been shown that this is sufficient for gossiping to succeed, even if this list is much smaller that the entire group. The membership protocol works as follows: Upon each gossip round, the identification of some locally known nodes is piggybacked on data messages. When a message is received, a node from the local list is picked at random and evicted, being replaced with the newly arrived node.

3.2 Buffer Management

The buffering layer at each node contains an independent queue for each possible destination. Messages enter the queues during each gossip round, as selected by the gossip mechanism. Even if the group is very large and contains hundreds of nodes, the small size of the local membership of each node ensures that only a few buffering queues are required at each node. Each message is inserted simultaneously in f distinct queues, allowing the space where the payload is stored to be shared, saving additional resources.

For short bursts of incoming messages, buffering alone is enough to spread the load in time and thus to avoid message losses. Buffering alone is also adequate for local area network in which congestion is transient. However, for continued loads, buffers are eventually exhausted. In sharp contrast with deterministic protocols, gossiping cannot be stopped until buffer space is available, in order not to allow a single perturbed node to degrade overall performance. The only option is thus to select a message to be discarded, either the newly arrived or one already in the buffer.

Dropping the newly arrived message is not an attractive option. When the system is congested it tends to result in the loss of entire gossip rounds as with no buffering at all. Such correlated loss quickly degrades atomicity guarantees. The goal is thus to avoid such loss at the expense of discarding other messages in the buffer. This is done by implementing a selection technique that combines, by order of preference, the following strategies: Semantic purging, age-based purging, and random purging. We now examine each of these in turn.

Random purging. A message is selected at random to make up room for each newly arrived messages. This ensures proper operation of the probabilistic protocol within the bounds of system capacity due to the intrinsic redundancy of gossip protocols [4] but is only marginally better when the system is congested for some time. It is however useful as a fall-back strategy.

Age-based purging. The message that has been relayed more times is discarded. The rationale for this is that those messages are likely to have reached more nodes and thus can be retransmitted from elsewhere [14]. This is the preferred purging strategy if no obsolete messages are discovered.

Semantic purging. A message that has been recognized as obsolete is discarded. By forcing correlated loss of obsolete messages, it is possible to avoid discarding non-obsolete messages. In contrast with other purging policies, in which message purging should be done lazily (i.e. messages should be discarded only if absolutely required), one can consider an eager strategy in which messages are discarded immediately when discovered to be obsolete, even if the reclaimed buffer space is not required. This has the potential advantage of reducing average buffer occupancy and thus of lower latency.

The main issue is how to convey the required semantics to the protocol while at the same time retaining its generality. It is thus not desirable that the protocol depends on the internal structure of messages, both to avoid the complexity of parsing the information as well as to allow a general purpose implementation of the protocol. This has been previously discussed in the context of semantically reliable multicast protocols [23] and the same implementation can be reused. Briefly, the proposed implementation is to associate a small bitmap to each message (e.g. 32 bits) and take advantage of the sequence number already present in protocol headers to remove duplicates and discover missing messages. If the i^{th} bit is set in the bitmap of message j, the protocol is informed that message with sequence number $j - i$ is considered obsolete. If the bitmap is all set to zero, the protocol defaults to the conservative approach of not considering message semantics.

This implementation allows that only obsolescence relations among messages from the same sender and close in the message stream are represented. This is however not a problem, as it is unlikely that two messages sent far apart can be found simultaneously in the same buffer and thus need to be compared. Obsolescence among messages from distinct senders can be represented indirectly using the following strategy: Consider a message m sent by some node p that makes a message m' by some other node q obsolete. Upon delivery of m' to p, the fact is noted. As soon as p has the opportunity of sending some other message m'', it uses it to mark the original m as obsolete. This implementation has the additional advantage of being efficiently manipulated at the protocol level. Namely, determining which messages are obsolete can be achieved with simple bit shift and logical operators [23].

Previous work on semantic reliability has also shown that the performance of semantic purging depends on the distance between related messages when compared to the amount of buffering. If related messages can only be found far apart in the message stream, it is unlikely that they can be found simultaneously in the same small buffer, thus reducing the likelihood of purging to occur. Notice however that it is possible to detect obsolete messages by observing all queues: If a message in one queue is made obsolete by another message in a different queue it is nonetheless considered obsolete. This makes the available buffer space for purging larger than each individual queue while reducing the impact on latency.

3.3 End-to-End Congestion Control

In contrast to previous proposals [2, 4], NEEM uses a connection oriented transport layer. More precisely, each node uses an individual outgoing TCP/IP connection for each node in its local membership list. When a network link is congested, the affected connections are throttled back forcing messages to accumulate in the corresponding queue at the buffering layer. If the bottleneck is the uplink, this happens simultaneously for all outgoing connections. If the bottleneck is elsewhere in the network, this has the additional advantage of relieving the uplink from traffic that would be discarded anyway. The resulting capacity can then be used by other competing connections.

This approach is also resource efficient, as the number of connections is as small as the local membership, and a reduced amount of buffering is used for each connection. Reducing resource usage is however not the primary goal when configuring each connection with a very small amount of buffer space. By reducing the total amount of buffering in the system, the latency can be dramatically reduced without severely impacting bandwidth usage [8]. This is critical, as the number of hops in gossip protocols multiplies end-to-end latency. Ideally, the buffer size would be dynamically adjusted by the operating system according to the current size of the window [8]. This mechanism is however not available in standard operating systems. The alternative is to statically calculate an approximate ideal buffer size given the expected bandwidth share and round-trip latency of connections. The rest is straightforward: `setsockopt()` is used to set the sender buffer to the desired value or, alternatively, an `ioctl()` system call used to poll current buffer usage.

Although using small buffers reduces the bandwidth usable by each individual connection, our approach does not prevent full usage of the already limited uplink bandwidth of each node. This happens because NEEM uses several connections sharing the limited bandwidth of the same uplink. A very small amount of buffering for each connection is thus enough to saturate its bandwidth share. In fact, the sum of buffers in all connections is as large as the default configuration of TCP/IP in most operating systems, which is enough to fully use the large bandwidth of local area networks or of high latency long distance connections over the Internet.

3.4 Discussion

Epidemic multicast protocols are usually described as requiring only a connection-less transport layer and in some cases requiring no explicit buffering. It is thus interesting to understand the role of the buffering and transport layers proposed. Notice that when the system is not congested, neither the buffers or the congestion control are used and thus we restrict our discussion to congested networks.

Besides the positive impact in competing traffic and overall network stability, using end-to-end congestion control at the transport level allows us to avoid loosing messages due to congestion within the network. In practice, this layer transforms a network with high bandwidth and high probability of message loss in a network with lower bandwidth but no message loss. The feedback provided to the upper layer about available bandwidth allows also to push buffering out of the transport layer as much as possible.

The buffer management layer uses purging to transform the low bandwidth and low loss network back into a high bandwidth and high loss network as the original network. The advantage is however in the quality of the loss. The original network discards bursts of possibly valuable messages thus failing to meet the assumptions on independent loss. The resulting transformed network is much smarter about message loss. It tries not to loose entire gossip rounds, at the expense of inducing correlated loss of already irrelevant messages.

Notice that each of the layers alone is of little use when dealing with network congestion. A smart message purging policy within the epidemic protocol is useless if the bulk of buffering is done elsewhere. Pushing message loss out of the network is also useless if full gossip rounds can still be lost due to a naive buffer management policy. The result of the combined use of these layers is the creation of a virtually ideal network for probabilistic multicast, which ensures adequate operating conditions to ensure atomic delivery of relevant messages.

4 Evaluation and Configuration

This section evaluates the performance of the NEEM protocol. In detail, we compare previously proposed buffer management policies (i.e. random and age-based) to semantic purging when combined with a connection oriented transport layer. This is done using traffic collected from Microsoft Flight Simulator 2002 (FS2002). We then evaluate the impact of traffic and system configuration parameters to determine the best configuration of the protocol.

4.1 Evaluation Criteria

The evaluation of the proposed system in the face of different application traffic profiles and purging configuration parameters requires the definition of a suitable performance metric. As noted before, NEEM does not aim at reducing the number of message losses when the network is congested. If the load exceeds the network capacity either the source is controlled, thus endangering throughput stability, or a number of message drops will inevitably occur. The purpose of our protocol is to create the conditions to drop more obsolete information than up-to-date information, thus improving the quality of the information provided to the users according to the semantics of the application and without endangering throughput stability.

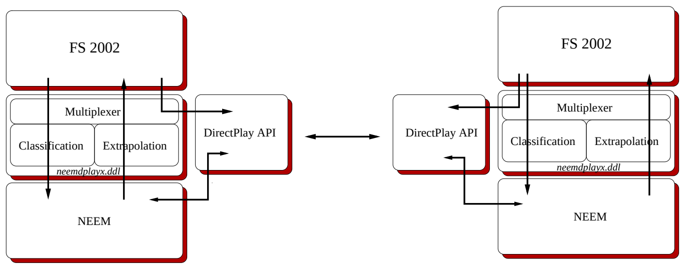
A good metric of the quality of the information delivered to the users in a system where a degree of message obsolescence exists is to count the loss of messages that never become obsolete. The primary evaluation criterion is thus that non-obsolete messages are atomically delivered according to the probabilistic protocols metric (i.e. either to almost all or to almost none recipients). Secondary criteria are reducing latency (i.e. messages are not arbitrarily delayed in order to allow purging) and that system configuration is robust (i.e. performance is stable despite variation of traffic and system parameters).

4.2 Integrating NEEM with FS2002

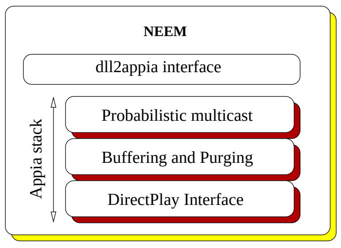
In order to have both a proof-of-concept prototype, and to ease the task of extracting realistic traffic patterns to evaluate the effectiveness of our approach, we have decided to experiment NEEM with the Microsoft Flight Simulator 2002 game.

The implementation exploits the fact that FS2002 is based on the DirectPlay API [17]. To intercept the call from FS2002 to the DirectPlay API we have implemented a new dynamic library that replaces the original `dplayx.dll` used by FS2002. The architecture of the new library, called `neemdplayx.dll`, is depicted in Figure 1(a). The library intercepts all data messages and forwards the remaining control operations to the origin DirectPlay implementation without further processing. Outbound traffic is classified, and obsolescence relations encoded in an abstract from which can be interpreted by NEEM below. Inbound traffic is pre-processed by an extrapolation layer. The extrapolation layer is responsible for storing the last position of each object and to compensate for lost or delayed messages in the probabilistic protocol, by extrapolating these messages from the past positions.

NEEM itself was implemented using the *Appia* protocol composition and execution framework [19]. The implementation is a modular composition of different layers, as depicted in Figure 1(b). The probabilistic multicast layer implements the epidemic dissemination of data messages. The buffering layer applies semantic purging to message queues according to the obsolescence relations identified by the `neemdplayx.dll`. Finally, the transport layer described in the previous section is implemented by interfacing back with the original DirectPlay library, which in turn calls the native TCP/IP implementation of the operating system. Since *Appia* is implemented in Java and the `neemdplayx.dll` is implemented in C++, a small adaption layer glues both components.



(a) DirectPlay interception



(b) NEEM

Figure 1: Implementation architecture.

Unfortunately, we do not have the resources to build a controlled experiment with hundreds of FS2002 observers. Therefore, we have used the implementation to extract realistic traffic pattern to feed simulations: the traffic pattern extracted from the prototype is synthesized and approximated during the simulations by a parametrized random generator. Traffic is collected from FS2002 by intercepting requests at the DirectPlay API to multisend messages. These are decoded [18] and logged. The following rule is then used to decide when each message becomes obsolete, if ever: If during a sequence of messages regarding the same airplane, the velocity vector differs from that of the first by less than a factor F, all but the last are considered to be obsoleted by its successors. This shows that even with $F = 1\%$ a share of 46% of all messages do become obsolete shortly after being sent. With $F = 2\%$ this share rises to 72%. This numbers disregards messages other than position updates, whose number is insignificant. The traffic pattern of FS2002 can be generalized as follows. Traffic is composed of an amount of traffic that never becomes obsolete. The remaining share r, of traffic is divided in d concurrent chains. Each message in the chain is made obsolete by its successor with probability l. The concrete traffic of FS2002 used fits in this model by considering $r = 1$, d as the number of aircrafts and $l = 0.72$ when $F = 2\%$.

4.3 Simulation Model

We use an high level event-based simulation model of the system, in which both the transport layer, the application and membership are abstracted. The epidemic protocol and the buffering layer are simulated in detail. The simulated system is composed of a configurable number of processes connected by a point-to-point network. Each connection is allocated a share of bandwidth using a token bucket algorithm and imposes a configurable latency on message transmission. Each node is connected only to a subset of distinct nodes chosen randomly which constitute its local membership. This membership is generated once before simulation runs and checked that every node is reachable from every other node in the system. This simulation model offers an interesting balance between accuracy and efficiency. Many of the data values shown in our results are obtained by binary search, which is computationally very intensive. A detailed simulation model of the transport layer, such as the one offered by ns-2 [20], would make our simulations prohibitively long without a significant improvement in accuracy, as previous work has shown that TCP/IP successfully approximates our model [8].

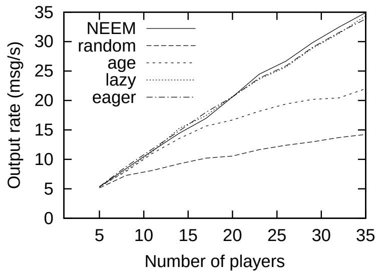
For the results presented in this paper, the network is configured such that each link has a 56 kbps bandwidth both for uplink and downlink and a latency of 25 ms. This is the worst case scenario in which all participants have limited bandwidth in face of the offered load. As the local membership size used is 12, this bandwidth is statically divided by the 12 outgoing connections. In practice, this corresponds to each node maintaining 12 outgoing TCP/IP connections and 12 incoming connections. We have used a number of nodes ranging from 50 to 500. As discussed in Section 2, even 50 nodes is out of reach of a simple multisend primitive and thus already an interesting application scenario. The default parameters used for simulations shown below are a buffer size of 10 messages per connection.

For the probabilistic multicast protocol we have used a fanout of 6 and maximum of 6 rounds per message. A justification for these values is outside the scope of this paper and can be found in [11]. We have verified experimentally that these values remain valid, but omit the results due to space constraints. The latency of the gossip mechanism is modeled as uniformly distributed between 0 and 25 ms, and is attributable mostly to scheduling latency. Each simulation is run for 300 s of simulated time. During simulation, events are logged to files and processed later to computed the desired statistics. The first 100 s and last 50 s are discarded to avoid transient states.

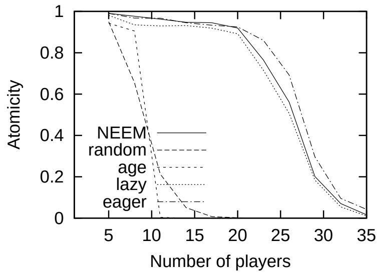
4.4 Simulation Results

Figure 2 depicts some relevant performance metrics of NEEM, namely the output rate, the atomicity of message delivery, and the latency of message delivery, as a function of the number of players. Values in this figure were obtained by running the simulation with a fixed number of 500 nodes, of which an increasing number of players, and recording the delivery of those messages that do never become obsolete.

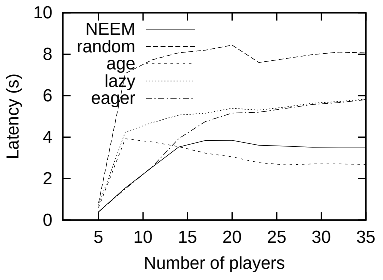
Figure 2(a) presents the average output rate of non-obsolete messages observed at a single spectator using a simple gossip protocol and NEEM. The effect of each purging strategy alone is also presented. Notice that only semantic purging strategies allow for a linear increase in the message rate. The curves depicted in this graph must be interpreted with care. With 25 players, pbcast is not “half as good” as semantic purging as the graphic shows. This reason for this is that different messages are discarded by different processes and “half of the atomicity” is not useful. A better metric is presented in Figure 2(b), which counts the number of messages which were delivered to more than 95% of processes. This means that semantic purging improves the usefulness of the protocol from 5 to 20 concurrent players. Latency was measured at a single process and only for delivered messages. As shown in Figure 2(c) random selection is the one which results in higher latency. Age based purging, on the other hand, results in low latency. Regarding semantic purging policies, it can be observed that an eager



(a) Output rate



(b) Atomicity



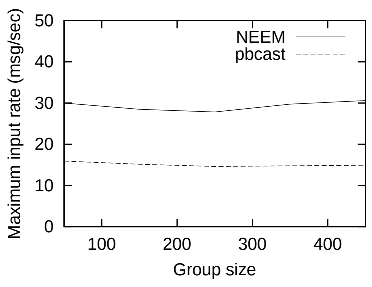
(c) Latency

Figure 2: Performance of the NEEM protocol.

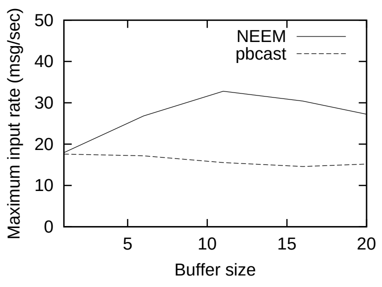
purging strategy results in better latency while the system is moderately congested, by eliminating transmission of as much obsolete messages as possible. Therefore, the eager semantic purging strategy is the best option, specially when combined with fall-back to age-based purging in NEEM.

Another way to assess the effectiveness of NEEM is to compute the maximum input rate that the protocol may sustain before breaking a given atomicity requirement. This metric is depicted in Figure 3. Maximum throughput graphics are calculated by binary searching (running a simulation for each combination of parameters of interest) of the input rate that causes the system to deliver at least 95% of non-obsolete messages to 98% of receivers.[2] We start by configuring the traffic generation with $l = 0.75$, $r = 0.75$, $d = 10$ and buffer size for 10 messages. Figure 3(a) shows that the results are independent of the total number of participants. This allows us to use relatively small groups of 50 participants for the remaining of this section thus greatly reducing the required simulation time. Figure 3(b) illustrates the impact of varying buffer size available at each node. We observe that there is a minimal buffer size that enables optimum throughput. Further increasing the buffer size presents no advantage for purging and would negatively impact latency. The amount of buffer required is intimately related with the amount of concurrent unrelated chains of messages. Fig-

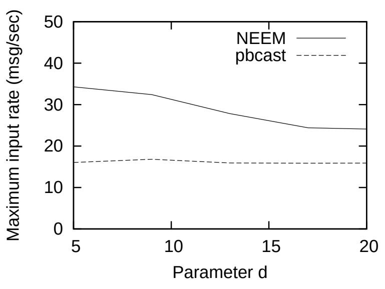
[2]As atomicity drops sharply, as shown in Figure 2b, this is very close to perfect atomicity. It is however less sensitive to variability when using binary search to compute the results than aiming for values closer to 100%.



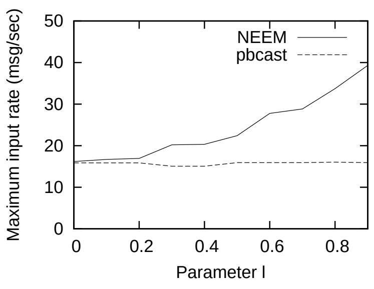
(a) Size of the group



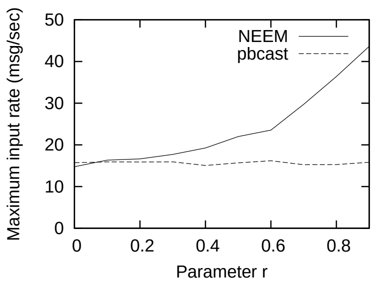
(b) Buffer size



(c) Traffic diversity



(d) Obsolescence chain length



(e) Amount of related traffic

Figure 3: Impact of group size, buffer size and traffic profile in maximum throughput.

ure 3(c) illustrates how the purging rate is affected by traffic diversity d by increasing the number of senders. Notice how increasing traffic diversity for a fixed buffer configuration decreases the purging performance. However, performance degradation is not sudden, ensuring that a small change in traffic characteristics does not result in a drastic change in the system output. Figure 3(d) presents the impact of increasing sequence lengths which cause increasing maximum achievable throughput. As this reduces the number of messages that never become obsolete it proportionally increases the amount of messages that can be purged (therefore, the system performance improves with the length of the obsolescence chains). Finally, Figure 3(e) presents the impact of sharing the channel with traffic that does not ever become obsolete, which naturally reduces the amount of traffic that can be purged (therefore, the system performs better with a larger amount of related traffic).

5 Related work

Our proposal stands on previous work on semantic reliability, epidemic dissemination and networking. We briefly discuss related work in each of these areas. Semantic reliability based on message obsolescence has been previously proposed in the context of group communication protocols and targeted at strong consistent replication [23]. Although the goal of our current proposal is different, a number of implementation mechanism can be reused, namely in managing message semantics and in purging buffers.

Epidemic dissemination protocols have been proposed for a variety of network settings. Recognizing that the probability of the message being delivered to all processes grows with the size of the initial set of processes receiving the message, it has been proposed that an initial optimistic multicast phase is used [2]. The epidemic phase is then done by negative acknowledgment thus saving network bandwidth in local area networks where a true multicast primitive is available. On the other hand, given knowledge about network topology it is possible to better spread network traffic across physical links, thus better coping with wide area networks [16, 10]. Although attractive, knowledge about the topology is difficult to achieve in the Internet and thus of limited applicability. By gossiping a fixed amount of messages with a fixed time interval, it is possible to bound the network bandwidth used by the protocol [4]. The protocol can be improved by preferring to discard messages which have been relayed more times [14]. In contrast to our proposal, messages are delayed by fixed period and not due to observation of network conditions, thus impacting latency even when the system is not congested. It has also been shown that a global deterministic view of the participant set is not required. Instead, it is sufficient that each participant keeps a random subset of the entire participant set computed dynamically with a fixed [4] or variable size [7]. We assume such a partial membership protocol to ensure the scalability of our buffering and transport layers.

A single other proposal has addressed the issue of controlling congestion in epidemic dissemination, although targeted at coping with insufficient buffering resources and not with competing network traffic. It has been proposed that the message input rate has to be adjusted to preserve reliability guarantees. This rate can be computed by the gossip protocol itself and thus dynamically adjusted to fit resource availability [25]. Our current proposal does not limit in any way the input rate.

Streaming media on the Internet has similar requirements to that of NEEM. Namely, TCP-friendly congestion control, low latency and uniformly distributed loss. It is thus interesting to consider for the transport layer some of the recent proposals in that area, such as DCCP [6]. Due to its widespread availability, TCP/IP is however still the best option, specifically, when configured for low latency [8].

6 Conclusions

Our proposal combines an epidemic multicast protocol with congestion control on the Internet and a sophisticated buffer management scheme. Although efficient gossip protocols have been previously proposed for local area networks, our current proposal is the first to support end-to-end congestion control and can therefore be safely used in the Internet (i.e. it is TCP-friendly). This is achieved without jeopardizing the throughput stability of the epidemic approach. A clever buffer management scheme that combines different techniques, including the use of knowledge about message semantics, ensures that available buffer space and bandwidth is applied as better fit for each application.

By combining the load-balancing features inherent to epidemic algorithms, with the network friendly operation of TCP, NEEM may simplify the task of traffic engineering in large-scale networks that must support event dissemination applications, such as large-scale publish-subscribe systems or distributed on-line games. As future work, we plan to study the interaction of epidemic algorithms such as NEEM with traffic engineering techniques.

The evaluation of the protocol is performed in two steps using a simulation model. First, traffic collected from Microsoft Flight Simulator 2002 is used to demonstrate the usefulness of the protocol, namely, in allowing hundreds of spectators. Then, synthetic traffic is used to explore boundary conditions and the behavior of the protocol in different environments and applications. This allows us to discuss how the protocol should be configured. A prototype of the protocol and its integration in Microsoft Flight Simulator 2002 has been implemented, illustrating the feasibility of the approach.

References

[1] K. Birman. A review of experiences with reliable multicast. *Software Practice and Experience*, 29(9), July 1999.

[2] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2), May 1999.

[3] S. Drucker, L. He, M. Cohen, C. Wong, and A. Gupta. Spectator games: A new entertainment modality of networked multiplayer games. Technical report, Microsoft Research, 2002.

[4] P. Eugster, R. Guerraoui, S. Handrukande, A.-M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. In *IEEE Intl. Conf. on Dependable Systems and Networks (DSN)*, 2001.

[5] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4), August 1999.

[6] S. Floyd, M. Handley, E. Kohler, and J. Padhye. Datagram congestion control protocol (DCCP). IETF Internet Draft, 2003.

[7] A.J. Ganesh, A.-M. Kermarrec, and L. Massoulie. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, February 2003.

[8] A. Goel, C. Krasic, K. Li, and J. Walpole. Supporting low latency TCP-based media streams. In *Intl. Ws. on Quality of Service (IWQoS'2002)*, 2002.

[9] K. Guo. *Scalable Message Stability Detection Protocols*. PhD thesis, Cornell University, Computer Science Department, May 1998.

[10] I. Gupta, A.-M. Kermarrec, and A. Ganesh. Efficient epidemic-style protocols for reliable and scalable multicast. In *IEEE Intl. Symp. Reliable Distributed Systems (SRDS)*, 2002.

[11] M. Hayden and K. Birman. Probabilistic broadcast. Technical Report TR96-1606, Cornell University, Computer Science, 1996.

[12] V. Jacobson. Congestion avoidance and control. *ACM SIGCOMM'88 Symp.*, 18(4), 1988.

[13] A.-M. Kermarrec, L. Masssouli, and A. Ganesh. Reliable probabilistic communication in large-scale information dissemination systems. Technical Report 2000-105, Microsoft Research, 2000.

[14] P. Kouznetsov, R. Guerraoui, S. Handurukande, and A.-M. Kermarrec. Reducing noise in gossip-based reliable broadcast. In *IEEE Symp. Reliable Distributed Systems (SRDS)*, 2001.

[15] J. Lin and S. Paul. RMTP: A reliable multicast transport protocol. In *IEEE Conf. Computer Communications (INFOCOM)*, 1996.

[16] M.-J. Lin and K. Marzullo. Directional gossip: Gossip in a wide area network. In *European Dependable Computing Conference (EDCC)*, 1999.

[17] Microsoft Corp. *DirectPlay 8.1*, 2002.

[18] Microsoft Corp. *Microsoft Flight Simulator 2002 Software Development Kit – Multiplayer/Flight instructor*, 2002.

[19] H. Miranda, A. Pinto, and L. Rodrigues. Appia, a flexible protocol kernel supporting multiple coordinated channels. In *IEEE Intl. Conf. Distributed Computing Systems (ICDCS)*, 2001.

[20] Network Simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[21] J. Pereira, R. Oliveira, L. Rodrigues, and A.-M. Kermarrec. Probabilistic semantically reliable multicast (extended abstract). In *IEEE Intl. Symp. Network Computing and Applications (NCA)*, 2001.

[22] J. Pereira, L. Rodrigues, and R. Oliveira. Semantically reliable multicast protocols. In *IEEE Symp. Reliable Distributed Systems (SRDS)*, 2000.

[23] J. Pereira, L. Rodrigues, and R. Oliveira. Semantically reliable multicast: Definition, implementation and performance evaluation. *IEEE Transactions on Computers, Special Issue on Reliable Distributed Systems*, 2003.

[24] R. Piantoni and C. Stancescu. Implementing the Swiss Exchange Trading System. In *IEEE Intl. Symp. Fault-Tolerant Computing (FTCS)*, 1997.

[25] L. Rodrigues, S.Handurukande, J. Pereira, R. Guerraoui, and A.-M. Kermarrec. Adaptive gossip-based broadcast. In *IEEE Intl. Conf. on Distributed Systems and Networks (DSN)*, 2003.