Performance Comparison of Reliable Multicast Protocols using the Network Simulator ns-2

Christoph Hänle
Vrije Universiteit Amsterdam
De Boelelaan 1081A,
1081 HV Amsterdam, Netherlands
chris@cs.vu.nl

Markus Hofmann
University of Karlsruhe[*]
Zirkel 2,
76128 Karlsruhe, Germany
hofmann@acm.org

Abstract

Reliable multicast protocols on top of the MBone are presently subject to intensive research. In the past, numerous protocols have been developed and their respective performance been analysed. Little progress has been made, though, to compare different approaches. In this paper, we use the network simulator ns-2 *to evaluate the performance of three protocols, namely* Scalable Reliable Multicast (SRM), Multicast File Transfer Protocol (MFTP) *and an enhanced version of the latter, called* Multicast File Transfer Protocol with Erasure Correction (MFTP/EC). *We also compare the results to each other and test the suitability for multicast file distribution.*

Keywords: Reliable Multicast, Network Simulator, Scalability

1 Introduction

Protocols for the reliable one-to-many data transfer can be constructed in various ways, and existing protocol architectures in fact use completely different techniques. As a consequence, they differ in bandwidth consumption and quality of service they offer to the application. Roughly speaking, reliable multicast protocols over the MBone all use IP's best effort multicast delivery service and provide mechanisms at least for error recovery, and possibly for flow control or congestion control as well.

The main difficulty is to cope with *scalability* and *heterogeneity*: The protocol should perform reasonably well even in large groups and for group members of greatly different Internet connectivity. These requirements are hard to encounter and the difficulties can not be completely hidden behind the interface to the application. For example, each receiving application might want to reliably receive packets in correct sequence and with minimum delay, and have the transmission rate be adapted to whatever rate it can handle. The sending application might require receivers to be kept in sync or might want to allow late joining. The ultimate protocol would achieve these service qualities using only little extra bandwidth for error recovery and would act fair to other network traffic. The criteria, however, are often competing aims and no single reliable multicast protocol architecture can meet them all simulataneously. Today, there is no *one-size-fits-all* protocol that can optimally serve the needs of all types of multicast applications. Instead, most multicast protocols are designed with stress on some criteria while neglecting others.

Application Level Framing (ALF) [3] has been proposed to help adapt transport-level services, such as reliability, to the needs of specific applications. ALF-based communication architectures are more flexible because they give the application more control over the end-to-end transmission. For example, ALF-based protocols normally do not try to put incoming packets back in order before handing them up to the application; they leave it to the application to deal with unsequenced packets in a clever way. Shifting the difficulty of coping with unordered packets to the applications is highly efficient, because the application can react in the most flexible way. This also means, however, that writing the application becomes more complicated [6]. In particular, the sending application has to partition data into units and to label each unit in an application context specific manner, such that out-of-order packets become meaningful enough for the receiving application to process them in a useful manner [7].

Another service quality that could be sacrified is quick loss recovery. For some types of applications such as multicast file distribution, there is no need for a receiver to

[*]Currently member of the Networking Software Research Department at Bell Laboratories, Holmdel, NJ, USA

quickly be provided with a repair packet; it is acceptable to leave space for the missing packet, wait until the end of the regular transmission and then fill the gap.

Because different multicast protocols often offer different service qualities, there is generally no superiority of one protocol over the other. By looking at parameters such as the number of duplicate packets processed, however, the extra cost can be calculated that one protocol requires for offering a better service than another protocol.

An issue of great importance is congestion control for reliable multicast. The Reliable Multicast Research Group of the IRTF is currently defining a set of reference simulations to assess different congestion conrol schemes [9]. While congestion control is fundamental to any transport-level multicast protocol, none of the protocols examined in this paper does yet integrate appropriate mechanisms. We expect this will change in the future.

At least three different ways exist to measure protocol performance, namely *analytical evaluation*, *experiments* and *simulations*. In the past, extensive work has been done to measure analytical performance [21, 1, 10]. Likewise, numerous results from experiments over the MBone have been reported [12, 15, 26, 8]. In general, analytical results can demonstrate well the implications on a protocol's performance when modifying parameters such as the number of receivers or their respective loss rates, because these parameters generally are inputs to the formulae. Analytical evaluation, however, only works for greatly simplified models of the network and background traffic and results therefore always remain questionable.

The approach frequently adopted is to run experiments over the MBone, so that the measured outcome is realistic by definition. This technique, however, does not allow to vary network load in an arbitrary way. Therefore, measurements reflect the protocol behavior in more or less typical cases, but performance evaluation under exceptional circumstances is hardly possible. Little can therefore be said about the protocol's robustness in atypical circumstances. Further, while re-running the experiment with a different multicast protocol, identical background traffic and network behavior is required for a fair comparison of protocol performance, and this can not be enforced in the MBone.

The way we took was to use a simulator that models the MBone in great detail. This technique lets us gain power over all parts of the network and yields the greatest insight. For example, we can easily set up a topology and vary network load arbitrarily while monitoring link loss rates. We can further place the multicast sender and the receivers anywhere within the network and measure the protocol's performance. It is simple to exchange one multicast protocol for the other and re-run the simulator, thus guaranteeing identical background traffic when drawing the protocol comparison. It is essential that the simulator models the MBone appropriately, or else the results become unrealistic. Related work on multicast protocol simulations include [14] and [13], which focuse on the impact of subgrouping and overlay structures on multicast performance, and [17], which introduces MESH and compares it to different approaches (centralized, tree-based and unstructured organization of group members).

The main motivation behind using a network simulator is that many multicast protocols greatly depend on temporal and spatial packet loss correlation. Getting these parameters wrong quickly leads to wrong conclusions. By simulating packet flows and losses in a realistic way, we expected the loss characteristics to be comparable to the MBone, given a preconfigured network load. We will investigate later in this paper whether or not this assumption holds. Using the simulation approach, the MBone's characteristics are not only modeled for typical network loads, but the behavior can even be extrapolated to atypical high or low traffic patterns. Simulations can therefore anticipate protocol behaviour in exceptional cases.

Our simulation models of the MBone have been used to compare the performance of three reliable multicast protocols to each other. Sections 2 and 3 introduce these protocols and the simulation environment, respectively. In Section 4, we compare the loss characteristics of the simulated network with traces taken from the MBone and give evidence that our simulation models are quite realistic. Section 5 measures bandwidth requirements for each protocol and draws a comparison. Also presented are measurements of SRM's packet loss recovery times. Section 6 gives a conclusion.

2 Reliable Multicast Protocols

Recently, a large number of reliable multicast protocols have been proposed [19]. In this paper, we compare three of them to each other, namely *Scalable Reliable Multicast (SRM)* [5], *Multicast File Transport Protocol (MFTP)* [18] and *Multicast File Transport Protocol with Erasure Correction (MFTP/EC)* [10], based on the results obtained from running tests with the network simulator ns-2. These protocols have been chosen because their source code has been available for integration into ns-2. Unfortunately, we have not been able to compare these receiver-initiated, NACK-based protocols to multicast protocols with subgroup-based error recovery. The only hierarchical, subgroup-based protocol featuring local error recovery and publicly available source code is the *Local Group based Multicast Protocol (LGMP)* [11, 13]. However, the highly multi-threaded implementation architecture of LGMP causes some major difficulties for the integration into the network simulator ns-2. Therefore, we have not yet been able to evaluate LGMP using ns-2. Nevertheless, LGMP has been analyzed using a different simulation tool [14] and extensive measurements on the worldwide MBone have been performed [12]. More work needs to be done to put these results in relation with the results presented in this paper. Tree-based multicast protocols, such as RMTP [15] or TMTP [27], have not been included in our simulations because we did not have any access to their respective source code.

The protocols included in our simulations all use a receiver-initiated error recovery scheme, but they differ in the way feedback traffic is reduced: SRM uses a slotting and damping technique. MFTP packs multiple NACKs into a single PDU, using one bit per data packet. Finally, MFTP/EC also packs multiple NACKs into a PDU, but further reduces NACK traffic because each bit reflects the status of a *group* of packets.

2.1 SRM

SRM [5] provides reliable many-to-many multicast packet delivery. That is, each participant can act as a sender and a receiver simultaneously. SRM is based on the ALF principle, that is, the protocol does not put received packets back in order (neither packets from a specific source nor packets from different sources). Rather, the application has to deal with unsequenced packets. Though SRM gives no timing guarantees, its design aims at quick packet loss recovery.

Lost packets are detected when gaps in the sequence number space occur. For this to work, SRM participants need a separate counter for every source. Detection works well unless losses occur immediately before the end of a packet sequence. SRM solves this problem by having all participants periodically disseminate low-rate *session messages*, each one containing the sender's own highest sequence number sent so far and the respective highest sequence number heard from every other known participant. Session messages are also needed for round-trip time measurements and to estimate the current number of participants.

To perform loss recovery, participants set a timer for every detected packet loss. If the timer expires, they transmit a NACK (SRM calls it *repair request*) to the whole multicast group. If participants receive a repair request before timer expiration, they cancel the timer and send no repair request instead (NACK suppression). Participants that receive a repair request for which they hold the corresponding data packet are called *responders*. They enter a repair phase for the data packet requested. Similar to sending a repair request, they set up a timer and transmit the data packet (SRM calls retransmitted packets *repair packets*) only when the timer has expired and no such repair packet has been received in the meanwhile. This mechanism is to avoid repair packet implosion. Like repair requests, repair packets are sent using global (unrestricted) multicast.

Crucial to SRM is the configuration of the timers, that is, the decision of how long to delay a request packet or a repair packet before eventually sending it. Floyd *et al.* [5] discuss this issue in detail. In essence, suppose that participant A sets a request timer to recover from a lost packet originating from sender S and assume that $d_{S,A}$ is A's current estimated propagation delay for the distance between A and S. Then A sets its timer to some random value in the interval $3^i \cdot [C_1 \cdot d_{S,A}, (C_1 + C_2) \cdot d_{S,A}]$ where C_1 and C_2 are variables that SRM continually adapts using past statistics of duplicate request detections. i dictates the number of backoffs, starting with 0: if A sends the request, it increments i and reschedules a request packet, using the broader interval to select the expiration time. This is necessary in order to cope with multiple losses. Further, if A receives a request packet prior to timer expiration and is not in a so-called *deaf period*, it cancels the timer, increments i and reschedules a new timer. A deaf period lasts half as long as the timer interval and is to avoid further backoffs due to duplicate requests. For details, see [5, 25].

A responder B — upon receiving a repair request — sets a timer to some value randomly chosen from the interval $[D_1 \cdot d_{A,B}, (D_1 + D_2) \cdot d_{A,B}]$, where $d_{A,B}$ is B's estimated distance to the requestor A and D_1 and D_2 are numbers that SRM continually adapts. There is no concept of backoff for repair timers. After sending or receiving a repair packet, however, B enters a deaf period of time $3d_{S,B}$ (S being the original sender or the first requestor) to avoid sending multiple repair packets for duplicate requests.

SRM senders send at some fixed rate, that is, there are no flow control or congestion control mechanisms.

2.2 MFTP

MFTP [18] is a protocol for the reliable file distribution to a large number of receivers (that is, up to the order of a few thousands) using multicast transmission. The protocol partitions the file into packets and builds equally-sized blocks of packets. Each block consists of as many packets as bits fit into an IP-packet of maximum size. For example, if the network can handle packets of at most 1500 bytes, then blocks cover roughly 12000 packets.

The sender initially multicasts all packets in a first pass. Receivers write all data to a file and leave appropriate space whenever they detect a packet loss. Later, they fill the gaps when the appropriate repair packet arrives. For each block with at least one missing packet, receivers send back in unicast mode a NACK-bitmap, reflecting the status (received/missed) of each data packet within the block. Receivers randomly delay the NACK transmission by up to 2s to reduce the implosion problem. The sender collects all NACK-packets, determines the set of data packets being requested at least once and retransmits those in a second pass. Again, at the end of the second pass, receivers send back NACK bitmaps. The procedure continues possibly with a third or forth pass, etc., until all receivers have completely received the file. Receivers leave the multicast group as soon as they have completed reception, causing the multicast tree to prune back.

Because the sender does not stop sending packets until the end of a pass, the protocol is insensitive to high round-trip delays (unless a pass becomes very short). Also note that error recovery does not start before all regular data has been sent. This does not pose a problem when packets are stored on disk, because disk seeks can arbitrarily access any file position.

Congestion control is performed by setting a threshold and having receivers drop out if their loss rate exceeds the threshold. In a future version, the sender will then set up a second session on a different channel and will send the data at a lower rate.

2.3 MFTP/EC

MFTP/EC was derived from MFTP and uses erasure correction methods to perform loss recovery more efficiently. Sender and receivers partition the set of data packets into groups of k packets each and build blocks of approximately 12000 groups. (k is some small integer and is discussed at the end of this section.) After a complete file transmission, receivers send back NACK-bitmaps, where each bit indicates whether or not all packets of a group have been received successfully. The sender multicasts in a second pass one redundancy packet for each NACK'ed group. Therefore, recovery passes in MFTP/EC only last up to a fraction of $1/k$ of a full MFTP pass (although in general, more passes than in MFTP will be needed).

Redundancy packets are computed by XOR'ing together a *subset* of the original packets for the requested group, according to a bit-vector $\mathbf{g} \in \mathbb{F}_2^k$. For example, if $\mathbf{g}$ is $(0,0,1,0,1,0,0,1)^T$, then the 3^{rd}, 5^{th} and 8^{th} original packet get XOR'ed to produce the redundancy packet.

Receivers, in turn, use the redundancy packets to repair lost packets. To do this, they keep track of which original and redundancy packets were received and gradually fill (for every group) a matrix $\mathbf{G} \in \mathbf{M}_{\mathbb{F}_2}(k \times k)$ by copying the vector $\mathbf{g}$ from the packet header into the next free column of the matrix.[1] A redundancy packet is rejected if either the matrix is already complete or if the new vector would be a linear combination of the vectors already stored the matrix. In both cases, the receiver discards the packet. A receiver has completed reception if the matrices for all groups are filled.

[1]Original packets are assigned a vector $\mathbf{g}$ too, namely the i-th original packet is assigned the i-th unit vector.

Let $\mathbf{s}_i \in \mathbb{F}_2^{12000}$, $i = 1, ..., k$ be the original data packets (for some group) in binary representation and let $\mathbf{c}_i \in \mathbb{F}_2^{12000}$, $i = 1, ..., k$ be the set of received (original and redundancy) packets. Then the following property holds:

$$(\mathbf{s}_1, ..., \mathbf{s}_k) \cdot \mathbf{G} = (\mathbf{c}_1, ..., \mathbf{c}_k)$$

As a final step, the receiver determines the original packets by solving the equation for the $\mathbf{s}_i$, which can be done because $\mathbf{G}$ is regular by construction. In effect, this again requires XOR'ing together a set of packets, this time redundancy packets though.

The number k of packets per group determines the efficiency of erasure correction. Higher values are generally feasible, because received redundancy packets become increasingly likely to repair a lost packet within the group it belongs to; they require, however, more processing power and disk activity, especially for the sender. See [10] for details. Lower values for k cause the gain of erasure correction to gradually vanish. At an extreme, if $k = 1$, then MFTP/EC effectively becomes MFTP. In our simulations k was always 16.

The sender's selection of $\mathbf{g}$ is also crucial to the protocol's efficiency and is performed before every new pass. $\mathbf{g}$ does not change within a pass. Badly chosen values for $\mathbf{g}$ will frequently result in linear dependencies. We found that codesets with a large Hamming distance are also feasible for the set of values that can be assigned to $\mathbf{g}$ in order to avoid dependencies. For details see [10]. We will measure how often linear dependencies occurred in Section 5.2.

3 Simulation environment

We have used the simulator ns-2 [25] from U.C. Berkeley/LBNL to simulate network and protocol behavior. In ns-2, the user defines arbitrary network topologies, composed of routers, links and shared media. Protocol instances can then be attached to nodes. A rich set of protocols is available. For example, there are several TCP flavors, UDP, SRM and RTP. Likewise, the user may choose between various types of applications. Among them are FTP, Telnet, and HTTP, which use TCP as the underlying transport protocol, and applications requiring a constant bit rate (CBR) traffic pattern, which use the UDP transport protocol. Multiple router policies can be configured, among them are drop-tail, early random detect and fair queuing. The routing model can be static or dynamic and there is support for multicast routing. The simulator is event driven and runs in a non-real-time fashion. Packet losses are simulated by buffer overflows in routers, which is also the dominant way packets get lost in the Internet. There is also support for error models other than losses through buffer overflows.

Tiers [4, 2] was used to create "realistic" network topologies in an automated way. Tiers generates a fully connected network made up of three different categories: One is the backbone or *WAN*, to which the *MANs* are attached. Routers that belong to MANs, in turn, are points of attachment for *LANs*. Each category consists of several nodes, connected by point-to-point links. There exist various configuration parameters for the size of the network and the connectivity. Based on the parameters, Tiers generates a random network topology. Tiers also assigns links a bandwidth based on its category and a signal propagation delay based on both the category and the distance between the two endpoints.

We have developed a graphical, interactive tool for setting up and supervising simulations based on the network editor Tkined [23]. Our tool helps the user to set up simulations quickly and intuitively and to monitor several parameters of the network or the protocol instances visually, such as packet losses, link utilization, queue sizes (per link) and number of received packets (per multicast protocol instance).

4 Network and Simulation Characteristics

In this section, we give evidence that the characteristics of our simulation models correspond fairly well to the characteristics of real networks. We compare the behavior of our simulated networks to the MBone traces from Handley [8] and Yajnik *et al.* [26]. Similar characteristics for modeled and real networks are essential when deriving conclusions from simulations. In particular, we examine packet loss rates and packet loss correlations (both spatial and temporal).

Let's start looking at the generation of random network topologies. Figure 1 depicts one of the topologies used in our simulations. All topologies were created by running Tiers with different configuration parameters.[2] The figure shows one WAN, consisting of 5 backbone nodes (depicted by a large square), which are connected by 6 backbone links in a redundant way. These WAN-links were assigned a bandwidth of 34368 kbps for each direction, thus representing an E3 carrier. The medium-sized nodes made up the 25 MAN networks, which consisted of 3 nodes each. Links between MAN nodes and those connecting a MAN to a WAN were given a bandwidth of 8448 kbps (E2 carrier). Finally, each of the 300 LANs consisted of 2 nodes, connected by a 10 Mbps link and attached to a MAN by an E1 carrier (2048 kbps).

All links were assigned a buffer for each direction, which was capable of queueing up to 50 packets for transmission. A queue size of 50 packets is also the ns-2 default value. Routing tables were computed prior to the start of the simulation using the distance vector multicast routing protocol (DVMRP) and did not change during the simulation. The routing policy was FIFO.

[2]The nodes were repositioned later by hand to achieve better readability, but the structure remained unchanged.

We defined background traffic for all topologies. In particular, TCP-Reno- and UDP-protocol instances were set up between two randomly elected nodes. To reflect the heuristic that servers are more often concentrated on single machines (think of an ftp server for example) whereas clients are usually spread all over the network, we have placed the servers (in one test scenario) on a randomly elected subset (60%) of all nodes. Server election was performed independently for TCP and UDP traffic.

4.1 Packet loss rates

We have monitored packet loss rates on a link basis. Simulation results show that hardly any packet was dropped when each node was assigned only few (say 1 to 5) TCP connections. Packet loss was below 1% on most bottleneck links while link usage was over 90%. This low loss rate certainly does not match well with the MBone. When more traffic was set up, loss rates increased. The labels on the links in Figure 1 show the average link loss rates for the scenario depicted in the right column of Table 1. Numbers are only shown for values $\geq$0.05%. If two numbers appear in the same label, then each one represents the loss rate for one direction. The average was taken over a period of ca. 450 simulated seconds. The figure shows link loss rates between 0% and 21.5%.

Moderate link loss rates (that is, $<$10%) occurred in most cases, even though 40000 TCP-Reno connections (running FTP instances) and 4000 UDP sources (running a CBR instances) were simulated on the 680 nodes, each protocol instance starting at a time chosen from the uniform distribution on the interval $[0, 300s]$ (for details, refer to Table 1). Obviously, it is not realistic to assume that the average number of TCP-connections originating from a host within only a few minutes is more than 60.

So why don't occur substantial packet losses unless there is an unrealistic high number of TCP connections active? First of all, it reveals the strength of TCP's congestion control scheme. Secondly, configuring 60 wide-area connections per host is unrealistic (except perhaps for servers), but assuming 60 outgoing connections per LAN or per intranetwork is much more pragmatic. It was mentioned above that we modeled LANs by two nodes only, which is probably not accurate in today's Internet. We also didn't model LAN traffic properly. If we view a single LAN-node as a representant of a whole LAN or even of an intranetwork and consider the outgoing connections to originate from some host within that LAN, then 60 TCP connections become much more realistic. Put it this way, the network grows considerably in size, because each LANs in Figure 1 then aggregates several nodes.

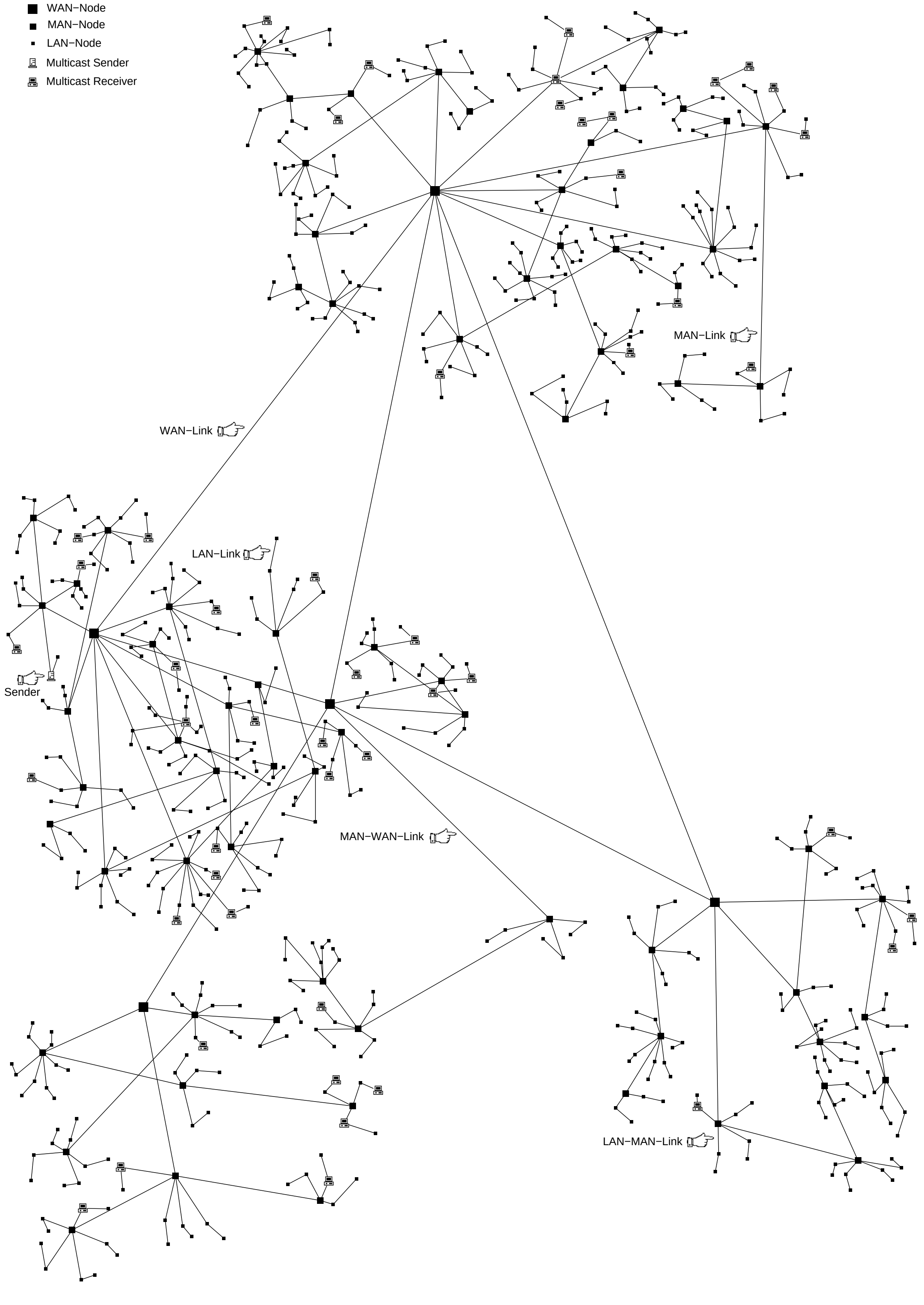


Figure 1. Topology for suite B (see Table 1). The numbers on the links represent the simulated loss rates when the multicast connection was set to MFTP/EC.

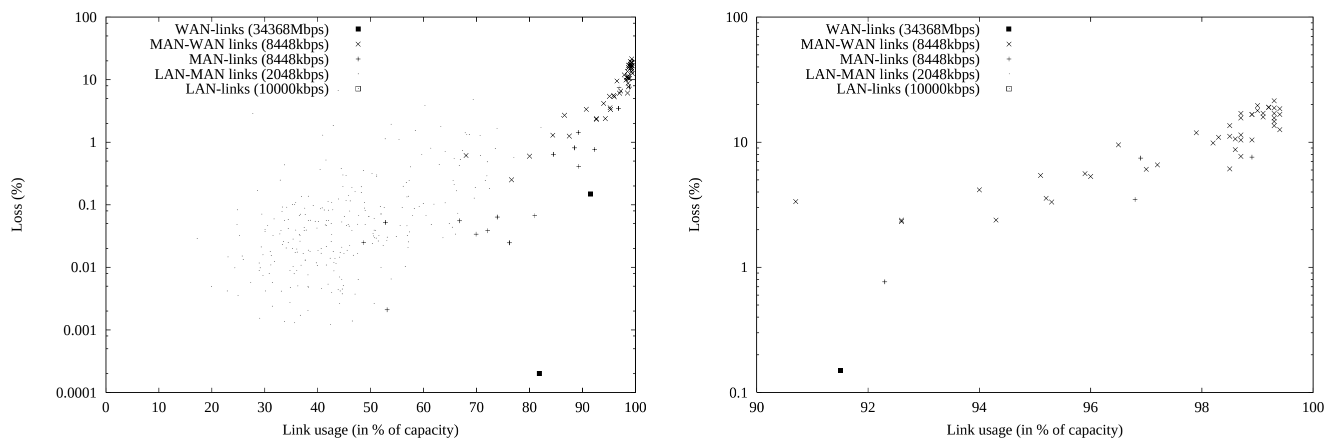


Figure 2. Link loss vs. usage for suite B. The right figure shows a part of the left figure in greater detail.

4.2 Spatial packet loss correlation

Let's look at the places where packets got lost in the MBone and where in our simulations.

Yajnik *et al.* [26] analyzed packet losses from MBone traces logged by audio multicast sessions over the MBone. The authors have used three different audio sources with up to 17 receivers that were spread all over the USA and Europe. Their measurements showed that only few packet losses occurred on the backbone links of the MBone. In general, the observed link loss rates ranged from 0.002% to 0.4% (with one notable exception of 6.6%)[3]. The rate at which packets got lost between the receivers and the backbone was much higher, ranging from 0.01% to 21.25%. Yajnik *et al.* further noted that very few packets were lost by only one receiver, that is, packets were unlikely to be dropped in the leaves of the network. In general, Yajnik *et al.* find that "spatially correlated loss in the network is low except for the loss next to the source".

Handley's MBone traces [8] are based on packet losses logged during multiple video sessions using the *vic video conferencing tool*. During the sessions, receivers were sending loss statistic reports to the monitoring workstation. Handley found that 50% of the receivers had a mean loss rate of 10% or lower, but that there were times when the loss rate was above 20% at 80% of the receivers and even that periods existed in which the loss rate was greater than 95% at 30% of the receivers. For most of the day, however, 25% of the receivers hardly lost any packets and the average loss rate was somewhere between 5% and 10%. 25% of the receivers experienced packet loss rates of more than 15% over hours. Handley also noticed that a small number of links are responsible for "larger scale correlation" of packet losses between receivers and that a large number of links existed with low loss rates, confirming that losses occur mainly spatially independently.

[3]Yajnik's multicast tree only depicts a graph constructed of *virtual* links, where virtual links aggregate a series of physical links.

For a comparison to our simulations, refer to Figure 2, which shows the *link utilization/packet loss* relationship of the simulation in Figure 1. A different marker is used for each type of link. Links that did not drop any packets are not shown. We observe that no single packet was lost on LANs. Moderate packet losses occurred on links that connected a LAN to a MAN. Packet losses ranged from 0% to 6.7%, a typical value being around 0.1%. MAN-links lost packets from 0% to 7.5%. Packets were lost on MAN-WAN connections at loss rates up to 21.4%. Finally, only two WAN links lost packets with loss rates of 0.15% and 0.0002%, respectively.

Essentially, packets were dropped mostly on MAN-WAN links in our simulations, i.e. in branches that lead to or away from the backbone. Further, the large number of links closer to the leaves of the network lost packets at low rates. These properties meet well the MBone traces and therefore, we expect quite realistic spatial packet loss correlations. We will consequently see in Section 5.3 in accordance with Yajnik that losses show some component for high correlation between receivers, but mainly still occur independently.

The figure also reveals a moderate relationship between link utilization and loss rate: The higher the utilization, the more frequent do losses occur (although this appears to be only a rough trend). Interestingly, different types of links seem to have a different loss-utilization relationship. For example, a loss rate of 0.07% occurred on LAN-MAN links when they were used around 30-60% to their capacity on the average. The same loss rate occurred on MAN-links when the usage was approximately 50-80% and for a WAN-link to produce 0.07% packet losses, traffic had to exceed ca. 80-90% of the link's capacity. Among the WAN links which are not represented in the figure (because they did not drop any packet), one exceeded 73.1% and another one exceeded 63.6% of the link's capacity. Why the network behaves this way and whether or not the MBone has similar properties is a matter for further research.

4.3 Temporal packet loss correlation

We now examine in how far packet losses tended to occur in bursts. To measure the temporal correlation of packet losses at receivers, a multicast sender was set up in our simulations. It disseminated test packets to a set of receivers. The receivers logged the reception of each packet. The logfile was later examined in order to derive the number of burst errors of length n, $n = 0, 1,$...[4]

Figure 3 shows the results of two scenarios that we have set up. For each burst of length n, the number of its occurrences is shown (as an aggregated value over all receivers). The straight lines in the graphs show how the curve would look like if losses were independently distributed in time. For example, if the average loss rate were q and N packets were sent, then burst errors of length $n = 5$ would be expected to occur $N \cdot (q^5 \cdot (1 - q))$ times for the average receiver. Note that the scale in the graphs is logarithmic. All graphs clearly show that packet losses are more likely to occur in bursts rather than being independently distributed in time.

Handley and Yajnik also observed that packet losses in the MBone show a trend towards longer bursts than what would be expected from an independent loss distribution. Handley also reports that the "base losses" were superimposed by burst losses of length 7 to 10 packets, which occurred every 30s and which contributed to the average loss rate to 1.5%. Yajnik *et al.* make a similar statement. They both assume that this phenomenon is the result of a bug in some routers that occurs during the time of processing route updates. The bug was not modeled in our static routing model and was also not recognized. Moreover, Yajnik detected some rare, but very long burst losses, covering up to several hundred packets, which could also not be observed in our simulations.

As a conclusion, the simulation yields a burst loss distribution that is comparable with traces taken from the MBone as far as the base loss is concerned.

[4]Counting burst losses of length $n = 0$ means counting pairs of consecutively received packets (plus one if the very first packet was received).

5 Protocol Performance Measurements

In this chapter, the performance of SRM, MFTP and MFTP/EC is evaluated.

We discuss two different test suites, referred to as *A* and *B*, each one consisting of three simulations with an identical network topology, identical background (that is, unicast) traffic and one multicast connection with comparable settings. The simulations within a test suite only differed in the *type* of the multicast connection, which was one of SRM, MFTP or MFTP/EC. Setting up different multicast protocols in otherwise identical environments allows for a fair comparison. Refer to Table 1 for the configuration details.

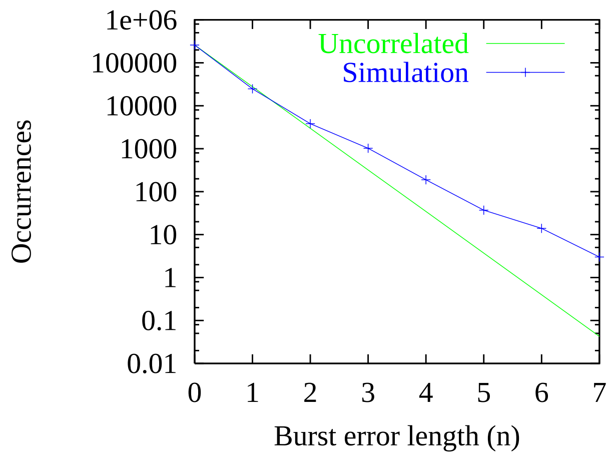
The focus was on one-to-many burst data transfer. Therefore, there was only one SRM sender of fresh data, even though SRM provides a many-to-many service. Of course, when it came to retransmissions, all participants could eventually become a sender of repair packets.

5.1 Receiver loss rates

Figure 4 shows the loss rates that each receiver has experienced. The upper figure belongs to test suite A. Because little background traffic was configured, packet loss rates were moderate. In particular, the average receiver loss rate was 11%, 10.6% and 10.7% for SRM, MFTP and MFTP/EC, respectively. The lower figure belongs to suite B, where we see high packet loss rates, due to higher background traffic. Here, the respective average loss rates were 36.6%, 22.8% and 23.1%.

The reason why receiver loss rates are very similar between MFTP and MFTP/EC receivers and quite different from SRM receivers in both test suites is because of the different ways the protocols work and because the generated multicast traffic affects the network's characteristics and therefore the loss rates. For example, both MFTP- and MFTP/EC-senders multicast data packets and receivers send back NACK-packets in unicast mode at the end of each pass. The generated traffic pattern is therefore quite similar, although the same receiver generally needs different times to complete reception. SRM's traffic pattern, however, looks very different: Request packets (NACKs) are *multicast* to the entire group rather than sent in unicast mode, and repair packets may originate from any participant, not just from the original sender. Moreover, SRM's loss recovery starts right upon loss detection. In contrast, recovery in MFTP and MFTP/EC is not before the end of the regular transmission. Finally, SRM participants send session messages every 2s, which is unique to SRM.[5]

[5]The period defaults to 1s in the ns-2 SRM implementation [25]. However, this period was considered too short, because the relationship between bandwidth used for session messages would almost have exceeded



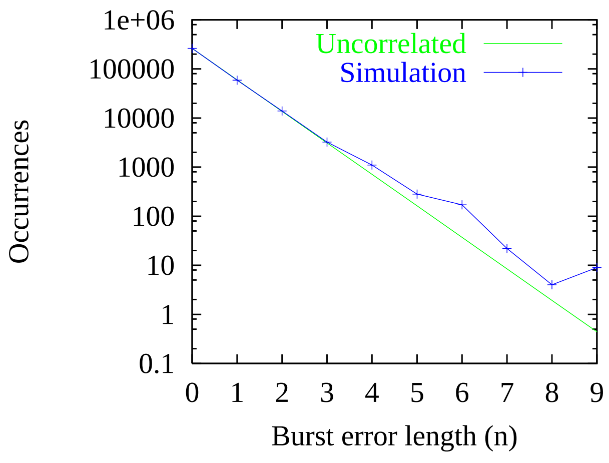


Figure 3. Distribution of burst errors for suites A (left) and B (right) with MFTP

In essence, we see slight differences in the receiver loss rates between MFTP and MFTP/EC and bigger differences between SRM and MFTP or MFTP/EC. The effect is visible in the upper figure. The reason why loss rates in the lower figure are generally much higher in SRM than in MFTP or MFTP/EC is because the current SRM implementation for ns-2 floods the network with repair packets substantially when loss rates are high, resulting in even higher loss rates due to additional traffic. We will further discuss this point below. However, SRM is a framework for reliable multicast that is still in progress. Recent proposals address the problems mentioned above by introducing a two-level hierarchy for session messages [24] and by using local error recovery, respectively [16]. Our simulation results clearly show the need for such enhanced mechanisms.

5.2 Bandwidth consumption

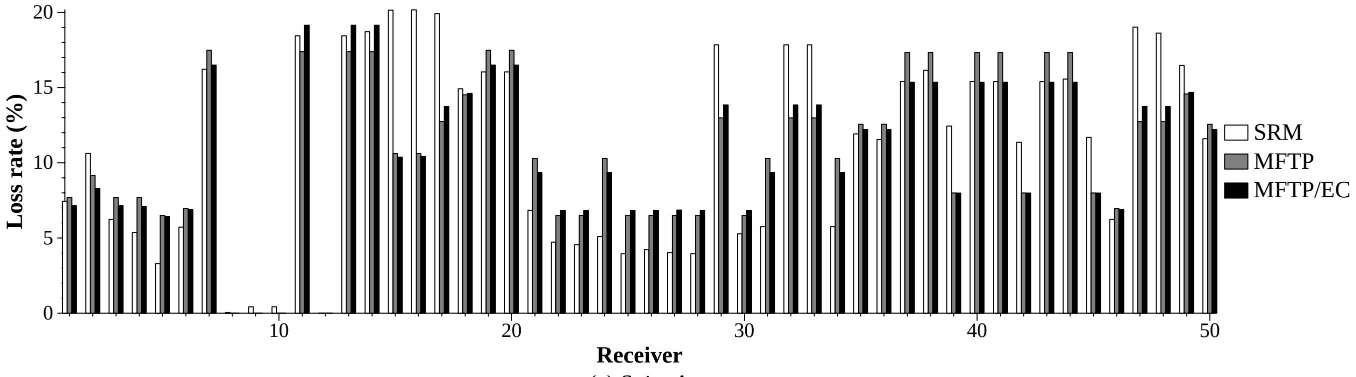
Let's take a look at bandwidth utilization. In all three protocols, data and repair packets are multicast to the whole group. It needs to be said that future revisions of SRM will probably include mechanisms to restrict retransmissions to a local scope. However, the current implementation addresses retransmissions to the whole group. Unlike SRM, MFTP's and MFTP/EC's repair phase starts at the end of the regular transmission, and receivers leave the multicast group as soon as they have completed the reception. Leaving the group causes the multicast tree to prune back, thus reducing unnecessary traffic. Therefore, to compare the cost of network utilization, we look at the *number of received packets* per receiver in each protocol. Using these statistics as an indicator for network utilization frees us from assessing and counting the cost for each individual packet flow within the network.

Receiver statistics also show evidence about which receivers suffer most from bad protocol performance, that is, which receivers must process a lot of *useless* packets. In SRM and MFTP, a useless packet is a double received packet. In MFTP/EC, a useless packet is one which transmission group is already complete or for which a linear dependence among the vectors $\mathbf{g}_i$ of the incoming and the already received packets arises (see Section 2.3). A packet is *useful* if it is not useless. For the following discussion, we neither consider repair requests in SRM nor NACK-packets in MFTP and MFTP/EC.

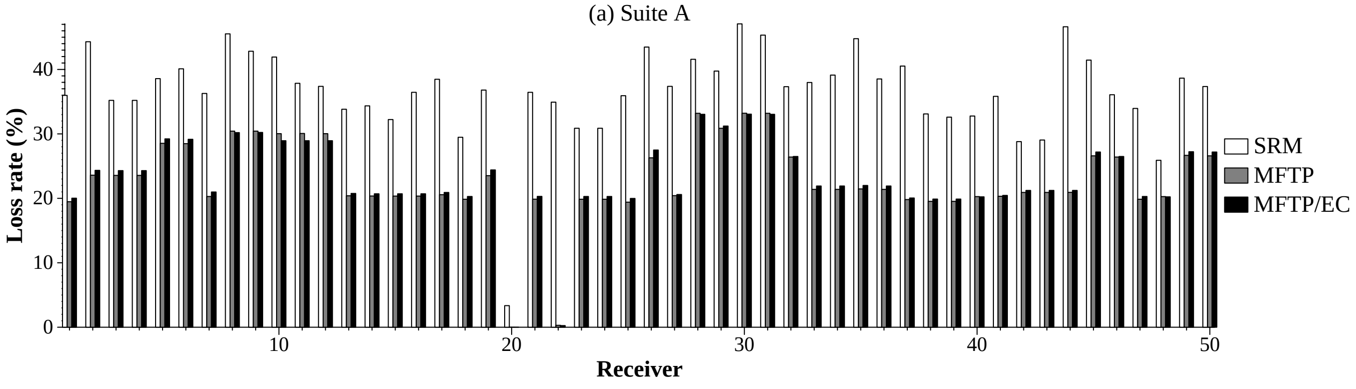
Figure 5 shows the result. Because the data transfer is reliable, all receivers have received the same number of useful packets, namely 4000 in both test suites. The six charts demonstrate that receivers with high loss rates showed a trend towards reception of only few useless packets. For example, receiver 30 in suite B experienced a high packet loss rate (see Figure 4 b) in all three protocols. Consequently, it received only few duplicate packets (see Figures 5 d,e,f), because a large number of missing packets meant that retransmissions often repaired a loss, and therefore did not count as useless packets.

For receivers with low loss rates, however, the number of useless packets is different in SRM as compared to MFTP or MFTP/EC. For example, receiver 20 in suite B experienced hardly any packet losses (see Figure 4 b). Figure 5 d demonstrates that this receiver had to process many double received packets with SRM while Figures 5 e,f show that it

the bandwidth used for data packets: A session message consists of $51 \cdot 4$ integers, that is, 816 byte (plus some header information) and gets multicast by each of the 51 participants to all group members. Therefore, in addition to 512 kbps data transmission rate, another 333 kbps would have been used for session messages. By doubling the interval, the session messages consumed only 166kbps of bandwidth. The shorter interval did not turn out to be a disadvantage to SRM in our tests.



(a) Suite A

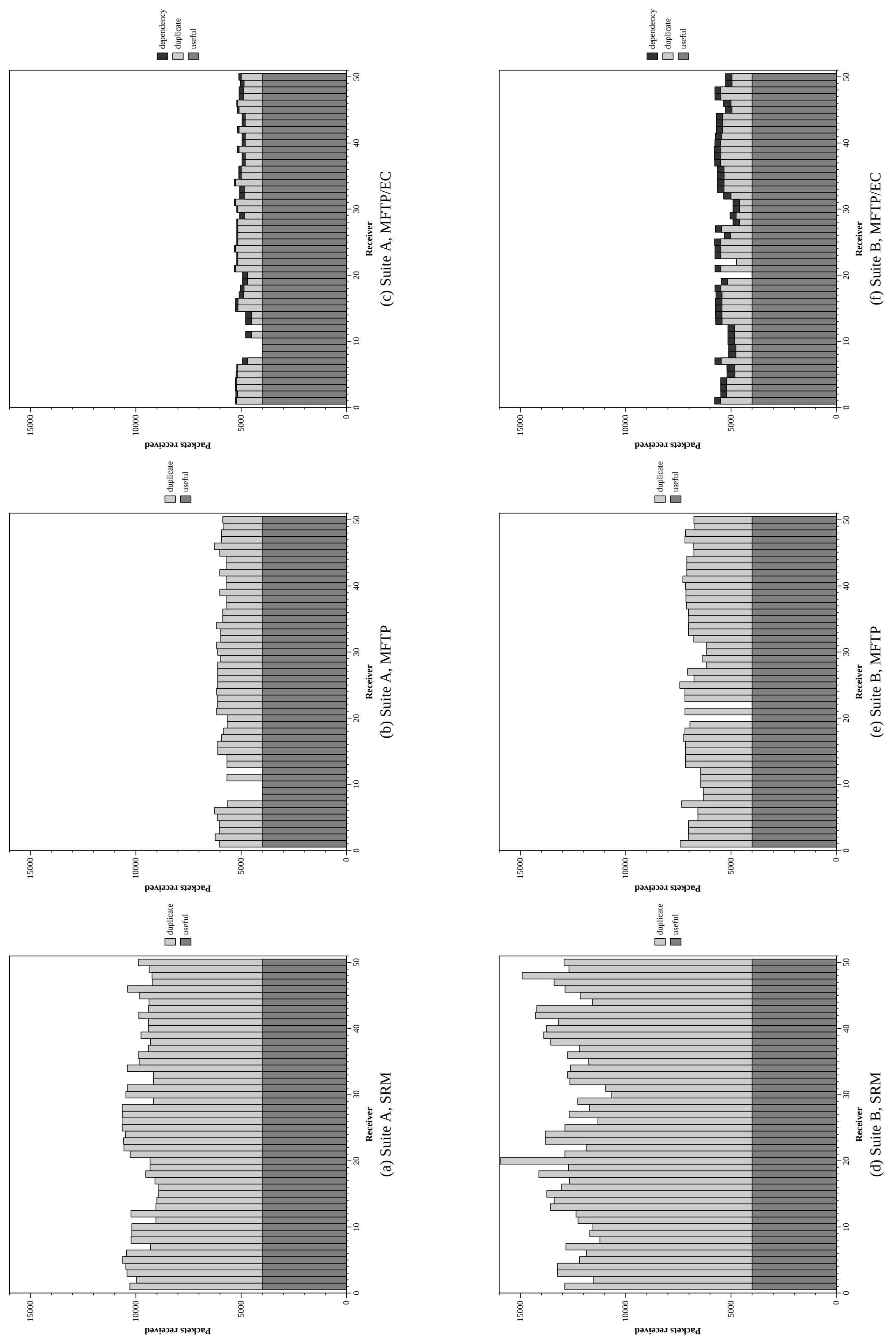


(b) Suite B

Figure 4. Receiver loss rates

	Suite A	Suite B
Topology	726 nodes (6 WAN, 60 MAN, 660 LAN) 727 links	680 nodes (5 WAN, 75 MAN, 600 LAN) 681 links
Background traffic	15000 TCP/FTP senders spread all over receivers spread all over 768 byte packet size 0...2048 kb file size 0...300 s start time	40000 TCP/FTP senders placed on 390 hosts receivers spread all over 1024 byte packet size 0...2048 kB file size 0...300 s start time
	2000 UDP/CBR senders spread all over receivers spread all over 1024 byte packet size 256 kbps transmission rate 1...100 s duration	4000 UDP/CBR senders placed on 390 hosts receivers spread all over 768 byte packet size 256 kbps transmission rate 1...100 s duration
Multicast traffic	One of SRM, MFTP, MFTP/EC 50 receivers 512 byte packet size 256 kbps transmission rate start time at 60 s 2000 kB file size Session messages frequency 2s (SRM) group size k=16 (MFTP/EC)	One of SRM, MFTP, MFTP/EC 50 receivers 512 byte packet size 256 kbps transmission rate start time at 60 s 2000 kB file size Session messages frequency 2s (SRM) group size k=16 (MFTP/EC)

Table 1. Test suite configuration details



(a) Suite A, SRM

(b) Suite A, MFTP

(c) Suite A, MFTP/EC

(d) Suite B, SRM

(e) Suite B, MFTP

(f) Suite B, MFTP/EC

Figure 5. Receiver statistics for test suites A and B

received no duplicate packet at all with MFTP or MFTP/EC. The reason is that in MFTP or MFTP/EC, receivers with few losses finish reception quickly. Once they are done, they leave the multicast group and skip the repair phase. In contrast, because SRM's loss recovery takes place during the regular data transfer, receivers have no way to escape repair traffic. These results clearly show the need for enhanced error correction schemes when doing error recovery on the fly. These schemes might either provide some kind of local error recovery [14, 15, 27] or might use an FEC-based scheme as proposed in [20].

Looking at the general outcome, Figure 5 illustrates that MFTP/EC receivers needed the fewest number of repair packets for loss recovery. MFTP did a good job, too. We also see that the number of packets useless to an MFTP/EC receiver due to a linear dependence has only a minor impact on the protocol's efficiency. Our XOR-based erasure correction scheme is less efficient than the Reed-Solomon erasure correction scheme described in [22], because packets becoming useless this way could have been avoided. For practical reasons, though, our XOR-based approach is still a good alternative, because it requires much less computation power for the necessary coding and decoding of packets.[6] SRM receivers needed by far the most repair packets. Where does this come from?

SRM can easily run into a situation in which multiple repair packets are multicast in response to a single retransmission request. SRM tries to avoid this undesirable situation by setting and adapting various parameters, but by design, the problem still remains.[7] In contrast, the MFTP or MFTP/EC sender always multicasts just one repair packet (unless it got lost another time). Therefore, it is not surprising that SRM causes more network traffic. In return, SRM receivers recover from losses during the data transfer, not afterwards. This allows to use SRM for applications others than a simple file transfer.

5.3 Packet statistics in SRM

This section focuses on the issue of duplicate transmissions of request and repair packets in SRM.

Figure 6 presents two charts, corresponding to suite A and B. There is a thin column for each data packet, consisting of three bars of different colors. The grey bar indicates how many receivers lost the packet; the lengths of the black bar and the light grey bar on top represent the number of repair requests and repair packets being sent, respectively. The packets were sorted in decreasing order by the number of receivers that detected a loss.

[6]The XOR-based scheme works approximately 5 times faster than the Reed-Solomon based scheme from [22] if packets are cached in memory (that is, if they need not be re-read from disk). See [10] for details.

[7]Indeed, there is a trade-off in SRM between duplicate packet flow and loss recovery speed. We discuss SRM's recovery performance later on.

In each figure, we identify two categories of lost data packets, namely packets that almost all receivers have lost and those that rarely got lost. For example, in the upper figure, 132 packets were lost by 46 to 48 receivers while the remaining 2968 packets were lost by 1 to 22 receivers. (The remaining 900 data packets were received by all participants). Clearly, because the former set of packets show highly spatial correlation, they got lost on an uplink near the multicast sender. The latter set of packets show little spatial loss correlation and therefore got lost on a downlink near the receivers. The sharp contrast is due to the absence of substantial packet losses on backbone links, which was also found to be characteristic for the MBone [26]. We call the former set of packets *frequently lost packets* and the latter set *scarcely lost packets*.

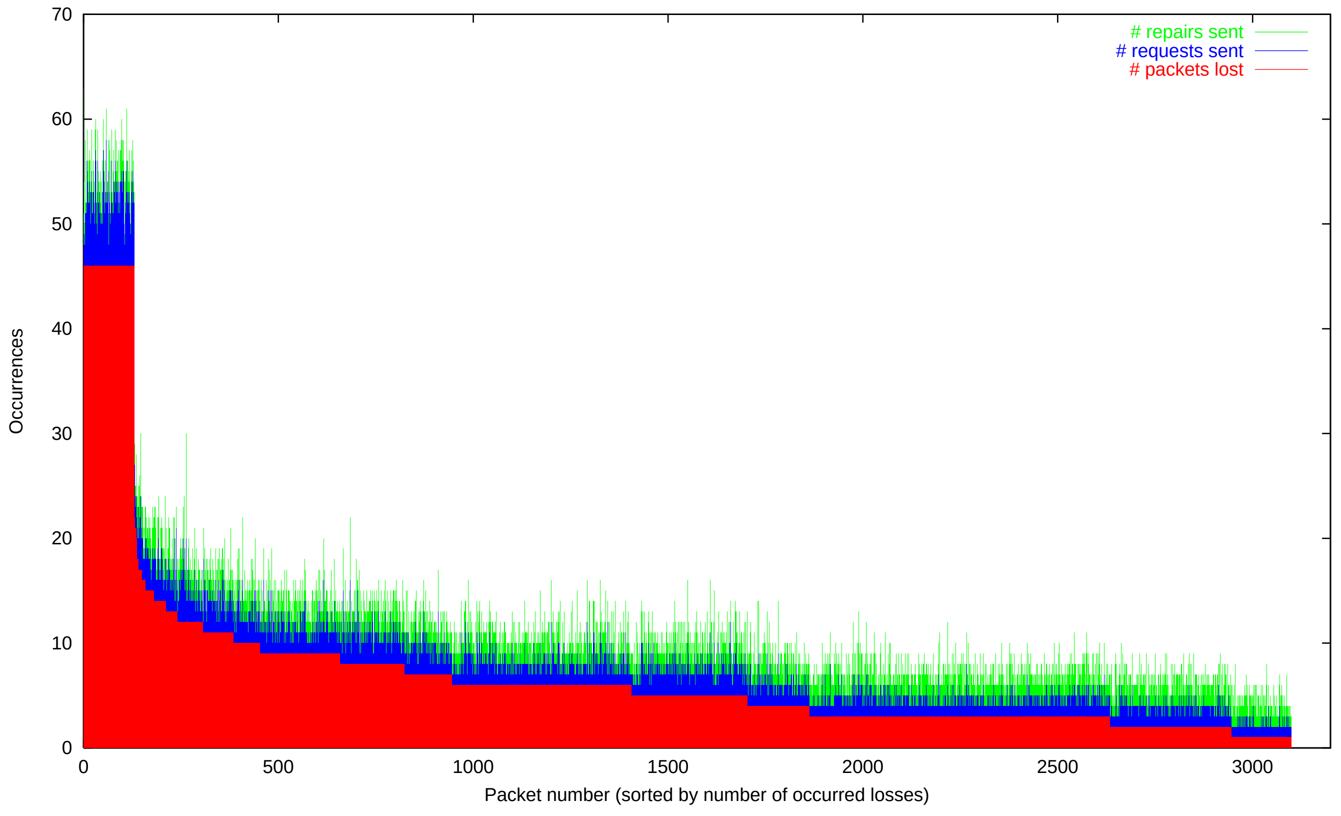
In the upper figure, for each frequently lost packet, repair requests were sent between 2 and 12 times, the average being 6.3. Each scarcely lost packets was requested between 1 and 8 times, the average was 2.1. The number of repair packets sent was between 1 and 7 times for the frequently lost packets (average: 2.5) and between 1 and 10 times for the scarcely lost packets (average: 2.3).

The values for the lower figure are higher, as expected: Repair requests were sent for the frequently lost packets between 1 and 18 times, 6.6 on the average and between 1 and 11 times for the scarcely lost packets (2.6 on the average). Each frequently lost packet was repaired between 1 and 11 times (average: 4.3) and each scarcely lost packet between 1 and 10 times (average: 3.4).

For both suites, we conclude that the values for the average number of repair requests differs between frequently lost packets and scarcely lost packets while the values for the average number of repair packets is similar. The reason is that the number of receivers which miss a data packet and schedule a request is usually low. Therefore, SRM can set tight bounds for the request timer interval (by adapting parameters C_1 and C_2) while still preventing duplicate repair requests reasonably well. Occasionally, however, a packet gets lost by the majority of the receivers, and then the request timers settings are too tight, causing a lot of duplicate repair requests.

In contrast, repair packets are scheduled upon reception of a request, and the set of potential responders is usually almost all group members. This causes SRM to widen the timer interval for transmission of repair packets (parameters D_1 and D_2). If occasionally, however, only few receivers receive the request packet, then the repair timers will certainly take longer than necessary to expire, but this will not introduce a high duplicate packet flow.

If request or repair packets get lost, then receivers will require additional rounds for loss recovery. They will send a second, third, etc. request packet. If this happened in the simulations, the additional request and repair packets were



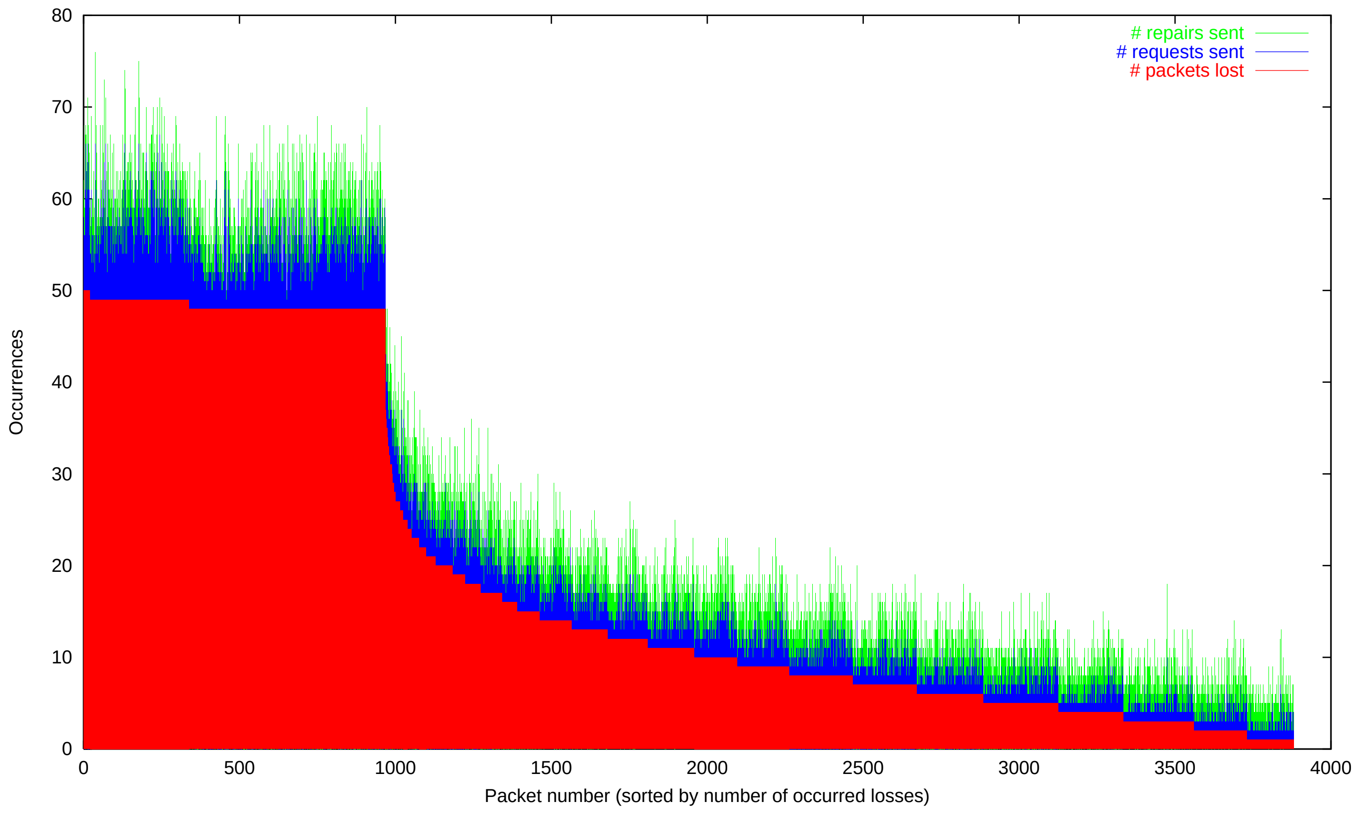


Figure 6. SRM's packet loss statistics for suites A (above) and B (below)

added to the ones from the first round to yield the figures.

As a conclusion, every lost packet triggered 2.27 request packets and 2.33 repair packets on average in suite A. Lost packets in suite B needed an average of 3.56 request packets and 3.62 repair packets for recovery.

5.4 Recovery speed in SRM

Until now, we have ignored the better service that SRM offers to the application: SRM hands up packets ”roughly” in order, as opposed to MFTP and MFTP/EC, which can completely scramble the proper sequence. For this reason, SRM is suitable for a wider range of applications.[8]

One example could be a multicast file transfer where data is read from or written to streaming devices such as tapes, that is, where data cannot be randomly accessed. SRM can also be deployed in real-time applications in which it is feasible to buffer incoming data for several seconds. For example, radio over the MBone is a candidate application for SRM: By buffering incoming packets, receivers have in many cases enough time to recover from lost packets before the packet's contents is used to play the sound. Therefore, the receivers can improve reception quality with SRM at the expense of a delayed playback.

Let's see how close SRM gets to a perfectly sequenced delivery. Have a look at Figure 7. Each graph shows the packet reception times (that is, either the time at which the original packet or a repair packet was first received) for three receivers with different loss rates. The higher a peek in the figure is, the longer it took the receiver to receive a repair packet. We have chosen the receivers with the lowest positive, the highest and some middle packet loss rate. Note that all receivers started reception at time 60s, though we have separated the curves for better readability by shifting them 20s up or down.

The upper graph shows good recovery for the receiver with a loss rate of 3.3%. Recovery worked worse for the other receivers: Although most of the time, packet recovery was quick, there are some rare, but high peeks of approximately 20s. Moreover, there was one exceptional high peek of 67.7s for packet number 3841, meaning that recovery took more than a minute.

The lower figure again shows that packet recovery was quick for the receiver with a low loss rate of 3.4%. It could almost seamlessly hand up packets in order. The receiver with 25.9% recovered from most lost packets within 20s, and the receiver with the highest loss rate, 47.1% got the last missing packet at time 172.9s, long after the regular data transfer. If the application needed packets in order, it would have to wait a long time and to buffer a lot of packets.

[8]Another benefit is that SRM provides a many-to-many multicast service, as opposed to MFTP and MFTP/EC, but we did not use this feature for our simulations.

6 Conclusion

With the appearance of a large variety of multicast protocols, appropriate methods for evaluation and assessment become more and more essential. While real network experiments in a large scale are pretty complicated, simple mathematical analysis is not sufficient for evaluation of complex multicast behavior. Network simulation is a promising approach, especially for large-scale multicast communication. However, it is essential to carefully design realistic network models that reflect real network behavior.

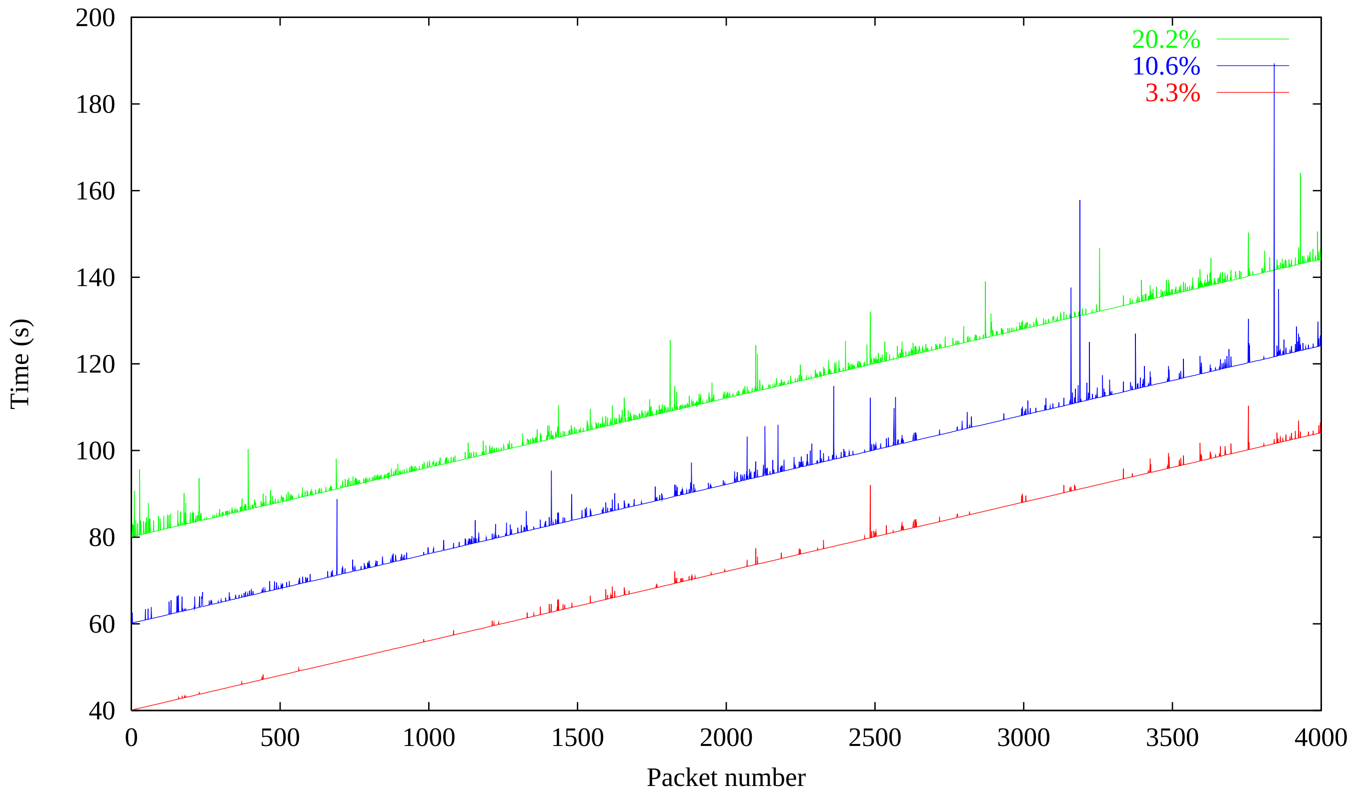
We have described the design and the implementation of a large-scale network model for multicast communication using the network simulator ns-2. We have focused on the provision of realistic network simulations which model the network and protocol instances in great detail. Network topology, link capacities, propagation delays and background traffic were chosen in a way that yielded roughly similar loss properties with the MBone.

The simulation models have been used to evaluate three example multicast protocols, namely SRM, MFTP, and MFTP/EC. These protocols were chosen because their source code has been available for integration into ns-2. We have run simulations for two different test suites, one with light and the other one with intensive background traffic.

The simulation results clearly show that multicasting repair packets to the entire global group can cause significant bandwidth usage. MFTP tries to treat this problem by simply delaying error recovery until the end of each pass. Receivers leave the multicast communication as soon as they are done, thus reducing reception of unnecessary repair packets. Network load could be further decreased by using a parity based error recovery scheme such as in MFTP/EC. Bandwidth usage of SRM has been quite high. However, we expect improvements by integrating local error recovery and hierarchical session message distribution into SRM.

The simplicity and the benefits of MFTP and MFTP/EC come for the cost of flexibility. Both protocols are restricted to file transfer applications, where data can randomly be accessed. (StarBurst has recently developed a special type of "streaming mode" service for MFTP, though.) Writing received data to streaming devices such as tapes causes additional problems. SRM is much more flexible in that it supports many-to-many communication and streaming-mode applications. In particular, SRM recovers from losses as soon as they are detected. SRM can therefore be deployed for interactive applications, especially if minor deviations from a perfectly sequenced packet stream can be tolerated.

Our simulations yield short loss recovery times for SRM, most often below 2s when loss rates were around 10%, and within 10s when loss rates were around 30%. However, some rare, but long recovery times occurred, longer than 60s. Those recovery times were due to multiple losses of



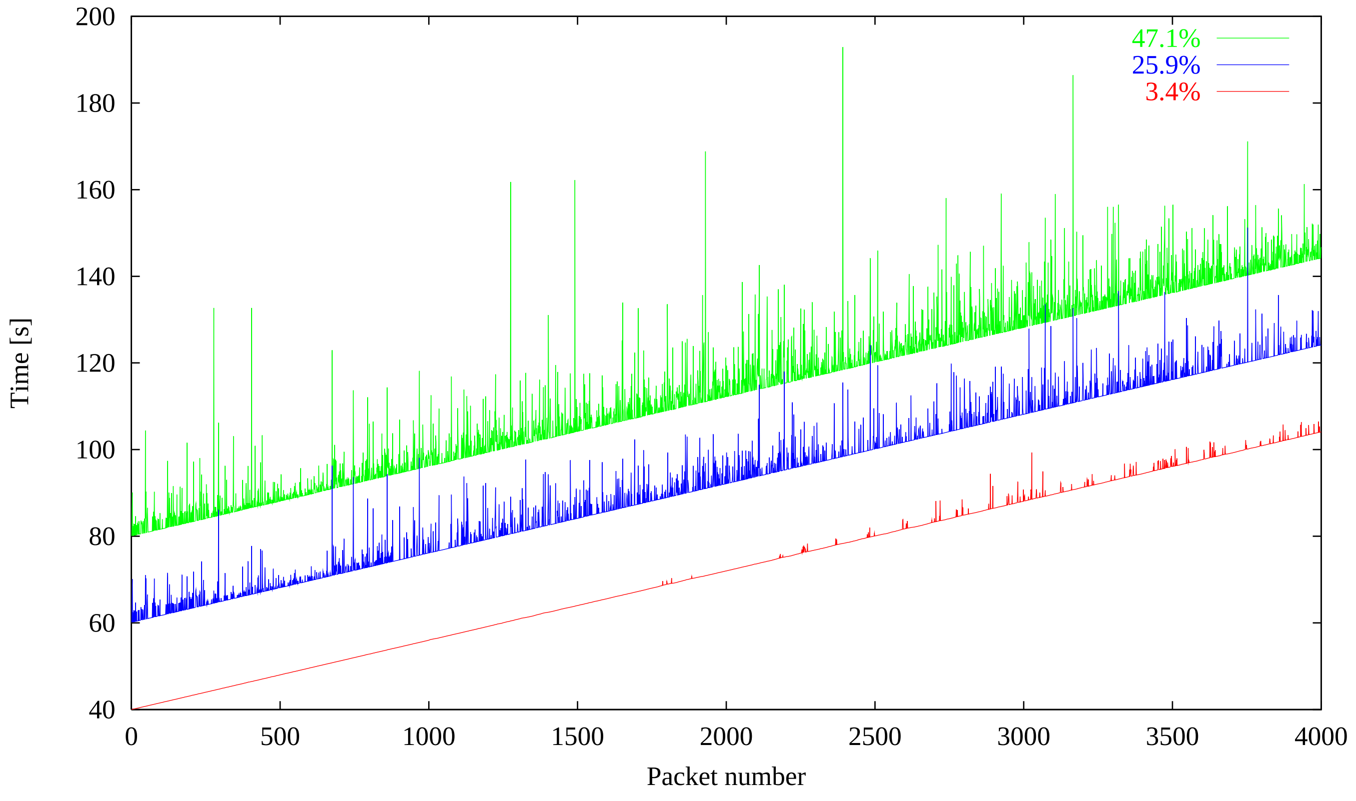


Figure 7. Packet recovery times for suites A (above) and B (below)

request or repair packets, causing receivers to backoff several times (for example, 3 or 4 backoffs were not unusual). If it is expensive for the application to conceal lost packets for a long time from the user, this might be a drawback.

We did not investigate scalability issues due to resource limitations: Some simulations had memory requirements of more than 1GB and ran for roughly one day on a 2x296Mhz Sparc Ultra. In larger sessions, we expect that NACK implosion becomes a problem for MFTP or MFTP/EC and session message traffic for SRM. In addition, more work needs to be done in order to compare these protocols to other approaches. In particular, it would be very interesting to simulate sub-group based and tree-based multicast protocols, none of which has yet been ported to ns-2. We hope that this will change in the future.

References

[1] M. H. Ammar. Improving the Throughput of Point-to-Multipoint ARQ Protocols Through Destination Set Splitting. Technical report, Georgia Institute of Technology, 1992. Also available as `http://mmlab.snu.ac.kr/~yslee/multicast/GIT-CC-92-13.ps.gz`.

[2] K. L. Calvert, M. B. Doar, and E. W. Zegura. Modeling Internet Topology. *IEEE Communications Magazine*, June 1997. Also available as `http://www.geocities.com/ResearchTriangle/3867/publications.html`.

[3] D. Clark and D. Tennenhouse. Architectural considerations for a New Generation of Protocols. In *Proceedings of ACM SIGCOMM*, pages 201–208, Sept 1990.

[4] M. B. Doar. A Better Model for Generating Test Networks. In *Proceedings of Globecom*, Nov 1996. Also available as `ftp://ftp.nexen.com/pub/papers`.

[5] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. In *ACM Sigcomm Conference*, pages 342–356, Aug 1995.

[6] M. Fuchs, C. Diot, T. Turletti, and M. Hofmann. A Framework for Reliable Multicast in the Internet. Technical Report RR-3363, INRIA, Sophia Antipolis, France, February 1998.

[7] M. Fuchs, C. Diot, T. Turletti, and M.Hofmann. A Naming Approach for ALF Design. In *Third International Workshop on High Performance Protocol Architectures (HIPPARCH)*, London, England, June 15-16, 1998.

[8] M. Handley. An Examination of MBone Performance. `http://north.east.isi.edu/~mjh/mbone.ps`, Jan 1997.

[9] M. Handley. Reference Simulations for Reliable Multicast Congestion Control Schemes. Draft Notes presented at IRTF RM Group Meeting, London, United Kingdom, July 6-7 1998.

[10] C. Hänle. A Comparison of Architecture and Performance between Reliable Multicast Protocols over the MBone. masters thesis, `http://www.cs.vu.nl/~chris/Thesis/Thesis.ps.gz`, Institut für Telematik, University of Karlsruhe, 1997.

[11] M. Hofmann. Homepage of the Local Group Concept (LGC). `http://www.telematik.informatik.uni-karlsruhe.de/~hofmann/LocalGroups.html`.

[12] M. Hofmann. *Worldwide MBone Experiments using LGMP*. Third IRTF RM Meeting, Orlando, FL, USA, February 22-24, 1998, Slides available at `http://www.east.isi.edu/rm/`.

[13] M. Hofmann. *Scalable Multicast Communication in Wide Area Networks*. Ph.D Thesis (in German), Institute of Telematics, University of Karlsruhe, Germany, Infix Publisher, Feb 1998.

[14] M. Hofmann and M.Rohrmüller. Impact of Virtual Group Structure on Multicast Performance. In A. Danthine and C. Diot, editors, *From Multimedia Services to Network Services*, number 1356 in Lecture Notes in Computer Science, pages 165–180. Springer Verlag, 1997. Proc. of Fourth International COST 237 Workshop, December 15-19, 1997, Lisboa, Portugal.

[15] J. C. Lin and S. Paul. RMTP: A Reliable Multicast Transport Protocol. In *Proceedings of IEEE INFOCOM*, pages 1414–1424, Mar 1996.

[16] C. Liu, D. Estrin, S. Shenker, and L. Zhang. Local error recovery in srm: Comparison of two approaches. Technical Report 97-648, USC Computer Science Department, January 1997.

[17] M. T. Lucas. *Efficient Data Distribution in Large-Scale Multicast Networks*. Ph.D Thesis, University of Virginia, May 1998.

[18] K. Miller, K. Robertson, A. Tweedly, and M. White. StarBurst Multicast File Transfer Protocol (MFTP) Specification. `draft-miller-mftp-spec-02.txt`, Jan 1997.

[19] MIST. Multicast Implementation Study. `http://www.tascnets.com/mist/`.

[20] J. Nonnenmacher, E. W. Biersack, and D. Towsley. Parity-based loss recovery for reliable multicast transmission. In *Proceedings of ACM SIGCOMM'97, Cannes, France*, 1997.

[21] S. Pingali, D. F. Towsley, and J. F. Kurose. A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols. In *SIGMETRICS*, pages 221–230, 1994.

[22] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *Computer Communication Review*, April 1997. Also available as `http://www.iet.unipi.it/~luigi/fec_ccr.ps.gz`.

[23] J. Schönwälder. Scotty, Tkined, Tnm. `http://wwwsnmp.cs.utwente.nl/~schoenw/scotty`.

[24] P. Sharma, D. Estrin, S. Floyd, and L. Zhang. Scalable Session Messages in SRM. Technical report, USC Computer Science Department, February 1998.

[25] UCB/LBNL/VINT. Network Simulator ns-2. `http://www-mash.cs.berkeley.edu/ns/`.

[26] M. Yajnik, J. Kurose, and D. Towsley. Packet Loss Correlation in the Mbone Multicast Network. In *Proceedings IEEE Global Internet Conference*, London, Nov 1996.

[27] R. Yavatkar, J. Griffioen, and M. Suda. A Reliable Dissemination Protocol for Interactive Collaborative Application. In *Proceedings of the ACM Multimedia '95 Conference*, November, 1995.