Packet Loss Recovery of Signaling Traffic in SCTP

Per Hurtig and Anna Brunstrom
Department of Computer Science
Karlstad University
Email: {per.hurtig, anna.brunstrom}@kau.se

Keywords: SCTP, signaling traffic, loss recovery, emulation

Abstract

The Stream Control Transmission Protocol (SCTP) was developed to be a viable solution for transportation of signaling traffic within IP-based networks. Signaling traffic is different from ordinary bulk traffic in many ways. One example of this is that the requirements of timely delivery usually are much stricter. However, the loss recovery mechanisms in SCTP are not fully optimized to these requirements. For instance, if packet loss occurs when the amount of outstanding data is small, a SCTP sender might be forced to rely on lengthy timeouts for loss recovery. This paper presents a number of proposals that try to solve this particular problem, with focus on the Early Retransmit mechanism. We propose a modification to Early Retransmit, to adapt it to signaling scenarios, and evaluate its performance experimentally. The results show that the modified Early Retransmit mechanism is able to provide significant reductions in loss recovery time. In some cases, the time needed to recover from packet loss was reduced with as much as 67%.

1. INTRODUCTION

The Transmission Control Protocol (TCP) [13] has traditionally been the most common protocol for transportation of data within packet switched networks. However, while a majority of network applications used today employ TCP, a growing number of applications have found the protocol to be too limiting for their specific needs. For example: TCP transmits byte streams, which means that applications must add their own record markings if they wish to encapsulate user messages; and applications using TCP can only receive data in the order it was sent, which can cause unnecessary delays if packets are lost or reordered.

One group of applications that benefits from having such limitations removed are applications that transfer signaling traffic. To better support this traffic, the Stream Control Transmission Protocol (SCTP) [19] was standardized by the Internet Engineering Task Force (IETF).

SCTP was originally developed to be a protocol for transportation of telephony (Signaling System No. 7 (SS7)) signaling messages. However, as the standardization of SCTP was completed and actual implementations started to emerge, it soon became subject to other types of signaling traffic as well. One example can be found in [15], where it is proposed that Session Initiation Protocol (SIP) [16] traffic is transported with SCTP. Despite the diversity in applications employing SCTP to transmit signaling data, some properties are common for traffic generated by signaling applications. For example, the amount of data transmitted is less than in ordinary TCP bulk transfers, flows are often bursty, and the requirements of timely delivery are usually much stricter.

Although SCTP is a protocol of its own, the specification is heavily influenced by TCP. For example, the loss recovery and congestion control are almost identical to those of TCP. There are several reasons for this, but one of the most important is that SCTP was designed to be TCP friendly. However, by inheriting the loss recovery and congestion control of TCP, SCTP also inherited some of the problems associated with these mechanisms. One problem for short, or bursty, flows can be found in the loss recovery mechanisms. SCTP uses two different mechanisms for loss recovery: retransmission timeout and fast retransmit. Retransmission timeout is a slow loss recovery mechanism, which is only used as a last resort. Fast retransmit, on the other hand, provides much quicker loss detection and is therefore the preferred recovery mechanism. The problem, however, is that fast retransmit does not work well, or at all, when the amount of outstanding data is small. Thus, if the amount of outstanding data is small, and packet loss occurs, a sender might be forced to rely on the slow loss recovery that retransmission timeout provides. This is troublesome for signaling traffic, as it typically consists of small amounts of data and have strict requirements on timely delivery.

Several proposals, for example Limited Transmit [2] and TCP Smart-Framing [11], try to solve the problem of slow loss recovery when the amount of outstanding data is small. However, in a signaling context both Limited Transmit and TCP Smart-Framing are inadequate, as further discussed in the next section, since they are only able to achieve limited improvement under certain circumstances.

Instead, we consider the *work in progress* proposal Early Retransmit [1]. We propose a slight modification to Early Retransmit, to better support signaling traffic, and present an experimental evaluation of its performance. To the best of our knowledge, this is the first evaluation of the proposal. The evaluation was conducted using the lksctp [5] implementation of SCTP. The experiments were executed in an emulated

 ISBN # 1-56555-317-9

network environment, using bursty traffic and network characteristics representative for signaling networks.

The results from the evaluation show that Early Retransmit can reduce the time needed for loss recovery significantly, when considering bursty traffic. For some packet losses the time needed to detect and recover from loss was reduced with as much as 67%.

The rest of this paper is structured as follows. In the next section we provide a detailed description of fast retransmit and its limitations in a signaling context. This section also describes Early Retransmit and the related proposals. The following section describes the experimental environment, and how the experiments were conducted. The next section presents the results we achieved, and the final section provides the conclusions of this paper.

2. PACKET LOSS RECOVERY IN SCTP

In this section, we describe the fast retransmit mechanism of SCTP. We also describe the two proposals Limited Transmit and TCP Smart-Framing, and explain why these are inadequate for improving the loss recovery of SCTP. Furthermore, this section also details the Early Retransmit proposal, and the non-bulk adaptation that we propose.

2.1. Fast Retransmit & Related Proposals

As previously mentioned, SCTP has two different mechanisms for loss recovery: retransmission timeout and fast retransmit. Fast retransmit is the preferred one, as it allows for the fastest loss detection. Fast retransmit is triggered by the reception of duplicate acknowledgments. Duplicate acknowledgments are sent by a receiver when it receives packets that are out-of-order. The reason for a packet to arrive out-of-order is either that packets have been reordered in the network, or that a packet has been lost, causing the following packets to be out-of-order. To disambiguate packet loss from reordering, SCTP requires that three duplicate acknowledgments are received before fast retransmit is invoked [18].

For example, if five packets, $S_1, \ldots, S_5$, are sent and S_2 is lost within the network, the receiver will generate duplicate acknowledgments for S_3, S_4, and S_5, causing the sender to fast retransmit S_2. While a threshold of three duplicate acknowledgments helps in preventing unnecessary retransmissions of reordered packets, it can also result in a performance problem. Consider the previous example of five packets. If packet S_1 or S_2 is lost, it will be possible to invoke fast retransmit as there are at least three packets following these. However, if one of the last packets is lost, the sender must rely on a costly retransmission timeout for loss recovery. In general, if the amount of outstanding packets are less than four, and no more data are ready for transmission, then it will not be possible to invoke fast retransmit at all. While this restriction is no serious limitation for applications that transmit large amounts of data, and thus keep a large number of packets in flight, it is a problem for signaling applications, as they transmit little, or small bursts, of data.

For example, consider a SCTP flow consisting of small bursts with signaling messages. Let us also assume that these bursts are separated in time by idle periods. If one of the messages is lost within the network it might be impossible to recover it with fast retransmit, especially if the gaps between different bursts are long, and the number of messages in a burst is small. The same problem of course also appears for short signaling flows. As signaling traffic has strict requirements on timely delivery, it is important that the SCTP loss recovery is enhanced to provide faster loss recovery in such situations.

As previously mentioned, both Limited Transmit (LT) [2] and TCP Smart-Framing (TCP-SF) [11] tries to enhance the loss recovery, when the amount of outstanding data is small. Even if these proposals target TCP, they are compatible with SCTP as well[1].

To enable timely loss recovery, LT allows a sender to send one previously unsent packet upon the reception of each of the first two duplicate acknowledgments. The intention of this strategy is to generate additional duplicate acknowledgments at the receiver when the sender is limited by its congestion window. However, there exists a number of situations where LT is not able to provide any improvements. For example, consider a sender that only has two packets to transmit. If one of these packets is lost, LT will be of no help. The reason for this is that no previously unsent packets, which can trigger duplicate acknowledgments, exist at the sender. In addition, LT will add at least one extra round-trip time to the loss recovery time, as it can not send any packets before receiving a duplicate acknowledgment.

The second proposal, TCP-SF, tries to enable fast retransmit by ensuring that at least four packets are outstanding all the time. To achieve this, TCP-SF is activated during the slow start phase and fragments packets that is about to be sent. For example, if a sender has no outstanding data and is about to transmit two packets, TCP-SF fragments these packets into four separate ones. In this way TCP-SF enables fast retransmit for the first outstanding packet. The benefit of this strategy is however limited, as it is only possible to recover the first of the four packets with fast retransmit. In addition to this limitation, TCP-SF is inappropriate for other reasons. For instance, it is incompatible with the message abstraction that SCTP uses, as it could unneccesarily split messages into separate packets. Regular SCTP implementations may split messages into separate packets, but only if a message is too large to fit in a single packet. Such large messages are, however, unlikely when considering signaling traffic.

[1] In fact, SCTP already uses an algorithm similar to Limited Transmit.

2.2. Early Retransmit

In [1], Allman *et al* introduce the Early Retransmit algorithm for both TCP and SCTP. To enhance the standard loss recovery of these transport protocols, when the amount of outstanding data is small, a dynamic lowering of the duplicate acknowledgment threshold is proposed. Based on the number of packets that are currently in flight, the threshold is reduced to support fast retransmission of the first outstanding data packet. For example, if three packets are in flight, the threshold is reduced to be able to trigger fast retransmit after receiving two duplicate acknowledgments. In general, if the amount of outstanding data (`ownd`) is less than four full-sized packets worth of data (`4*SMSS`), and either no unsent data is ready for transmission or the receiver-advertised window does not permit any more packets to be transmitted, then fast retransmit is permitted when `ownd-SMSS` bytes have been acknowledged. Therefore, Early Retransmit will permit fast retransmit of a single lost packet, as long as there are at least one packet left that can generate a duplicate acknowledgment.

Compared to the previously discussed proposals, LT and TCP-SF, this approach allows fast retransmit to be invoked in nearly all situations. The only situations in which Early Retransmit is of no help, is when multiple packets are lost, or if a single packet is sent and lost.

A possible problem with Early Retransmit, discussed in [1], is that a reduction of the duplicate acknowledgment threshold could make SCTP less robust to network reordering. As mentioned in the previous subsection, two different events can cause duplicate acknowledgment generation at a receiver: packet loss or network reordering. If reordering occurs, and the degree of reordering exceeds the value of the duplicate acknowledgment threshold, SCTP will spuriously retransmit packets. Several researchers (e.g. [9][8]) have concluded that the prevalence of network reordering could be high in some Internet paths, and that this could be harmful to transport protocol performance. However, a number of suggestions exist that could reduce the impact of network reordering. For example, Allman *et al* [1] suggest that DSACKs [4] are used to detect spurious retransmissions, and limit the use of Early Retransmit upon such detection. Network reordering may, however, not be an issue in most signaling scenarios, as networks used for this kind of traffic often are managed and therefore less prone to reorder packets.

Considering signaling traffic, a more serious problem with Early Retransmit, as specified by Allman *et al* [1], is that it is not appropriately adapted to non-bulk transfers. This is a significant limitation, as signaling traffic typically is non-bulk. In the next subsection we explain why such an adaptation is necessary, and also propose one.

RTO_{init} (ms)	200	
RTO_{min} (ms)	100	
RTO_{max} (ms)	400	
$SACK_{delay}$ (ms)	40	
Burst size (messages)	4	7
Inter-burst gap (ms)	200	100
Bottleneck bandwidth (Kbit/s)	500, 1000, 2000	
One way end-to-end delay (ms)	5, 10, 15, 20, 25	

Table 1. Experimental Parameters

2.3. Modified Early Retransmit

As previously mentioned, Early Retransmit is intended for both TCP and SCTP, and the actual algorithm is exactly the same for both protocols. We also stated that this was inappropriate, as it does not work well for non-bulk traffic. For instance, consider an application sending five SCTP signaling messages with a payload of 200 bytes each. Further, let us assume that the application does not send them at once, but rather with a small amount of time in between. What will happen is that these messages are not bundled into larger packets, but rather sent in five different packets. If the Early Retransmit algorithm is used as specified in the previous subsection, fast retransmit can never be triggered. Since the amount of outstanding data (`ownd`) will be less than one `SMSS`, assuming an `SMSS` larger than 1000 bytes, `ownd-SMSS` can not be acknowledged.

To better support such scenarios, we propose a modification to the SCTP version of Early Retransmit. This modification changes the requirements of Early Retransmit to work on a *number of packets* basis, instead of a *number of bytes* basis. With this modification, the requirements of Early Retransmit becomes: if the amount of outstanding packets (`opkt`) is less than four, and either no unsent data is ready for transmission or the receiver-advertised window does not permit any more packets to be transmitted, then fast retransmit is permitted when `opkt-1` packets have been acknowledged.

In our implementation of Early Retransmit, evaluated in the rest of the paper, this modification is used.

3. EXPERIMENTAL SETUP

To evaluate Early Retransmit in a signaling context, we modified the standard parameterization of SCTP to conform with the high requirements that signaling traffic usually has [12]. These values can be found in Table 1, and are based on recommendations from the telecommunication industry. The values are much lower than the values in the SCTP RFC [19], in order to provide the timeliness required in signaling environments. To further support timely delivery, signaling environments are often designed to have low end-to-end delays, along with dedicated bandwidth for signaling traffic. In the evaluation we have used delay and also bandwidth character-

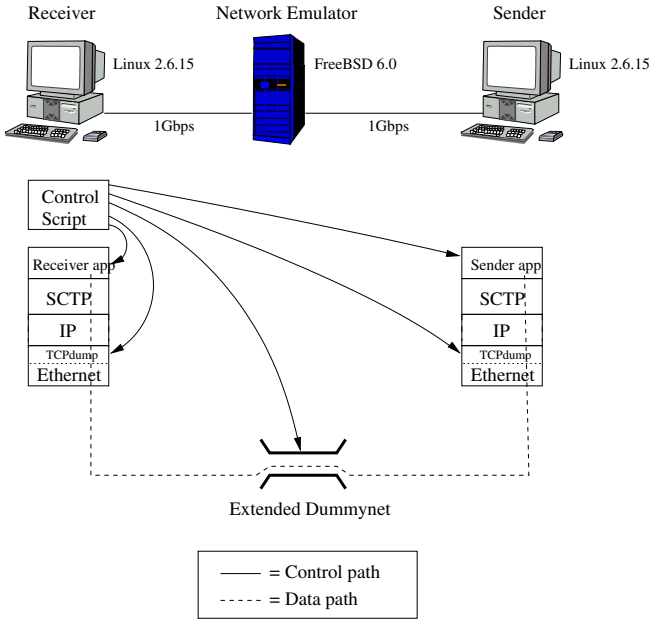

Figure 1. Experimental environment

istics that are relevant to such environments. These parameters are listed in Table 1 as well.

As noted earlier, signaling traffic is different from ordinary TCP bulk traffic, presumably bursty in the case of SS7 signaling [3][17]. Therefore, a variety of bursty traffic patterns were used in the evaluation. The following patterns were considered: (i) static sized bursts with static inter-burst gaps; (ii) static sized bursts with exponentially distributed inter-burst gaps; and (iii) random sized bursts with static inter-burst gaps. For traffic pattern (i), two different burst sizes and inter-burst gaps were used. These are listed in Table 1. For pattern (ii) the same values were used, but for this pattern, the inter-burst gaps served as mean values in an exponential distribution. Traffic pattern (iii) was also created using these values, but for this pattern the different burst sizes were used as mean values in discrete uniform distributions. For both of these mean values (n), the ranges of the uniform distributions were $[n-3, n+3]$.

Let us now consider how the experiments were conducted. The environment used for the evaluation consisted of three ordinary hosts that acted as sender, receiver, and network emulator. The sender and receiver were instructed to communicate via the network emulator, which delayed and dropped incoming packets as instructed. The entire environment is shown in Figure 1. Since we evaluated the loss recovery performance of a standard SCTP implementation and an Early Retransmit extended SCTP version, packet loss played a central role. To achieve precise control over packet loss the network emulator was equipped with an extended version of the Dummynet network emulation software [14]. The extended version [7] allows precise control over packet loss with the possibility to specify these on a per packet basis, using precomputed patterns.

Using this environment, the experiments were conducted as follows. The receiver initiated an association with the sender, which responded by transmitting a fixed number of bursts according to one of the previously mentioned traffic patterns. When this traffic then passed through the network emulator, a single packet from the flow was dropped by the emulator. The time needed for the lost packet to be received by the receiver was then measured. Using this method, the whole procedure was repeated[2] until packets in all the different positions of the flow had been subjected to loss. The metric used to evaluate Early Retransmit was the message transfer time (MTT). We define MTT as the time required for a message to travel from the sender to the receiver application. When a message was lost within the emulated network, the time needed for loss detection and retransmission was, of course, included in the MTT.

For the static traffic pattern (i), three replications of each experiment were made and the median value is reported. As both the bursts and the inter-burst gaps were static, three replications were enough to eliminate possible random outliers. For the patterns including randomness, (ii) and (iii), more replication were of course needed to provide statistical validity. For these traffic patterns, we conducted 40 replications of each experiment and reported the mean values together with the 95% confidence intervals.

4. RESULTS

In the following three subsections, the results for the different traffic scenarios are provided.

4.1. Static Burst Sizes & Static Inter-burst Gaps

In Figure 2, a representative example of the results for static burst sizes and static inter-burst gaps is shown. The y-axis shows the MTT of individual SCTP messages. That is, how long did it take for a message to travel from the sender to the receiver application. The x-axis shows the transaction sequence number (TSN) of the corresponding message. For the experiments shown in this graph ten bursts, each containing four messages, were measured[3]. For these experiments, the bottleneck bandwidth was 2000 Kbit/s, the delay 15 ms, and the time between each burst 200 ms.

In the graph we can see three different data sets. The first set, labeled “No Loss”, shows the MTT of the individual mes-

[2]Including association establishment, transmission of new packets, and association termination.

[3]Actually 12 bursts were transmitted but the first two bursts were omitted to avoid interactions with the association initiation. The same procedure was used for all experiments described in this paper.

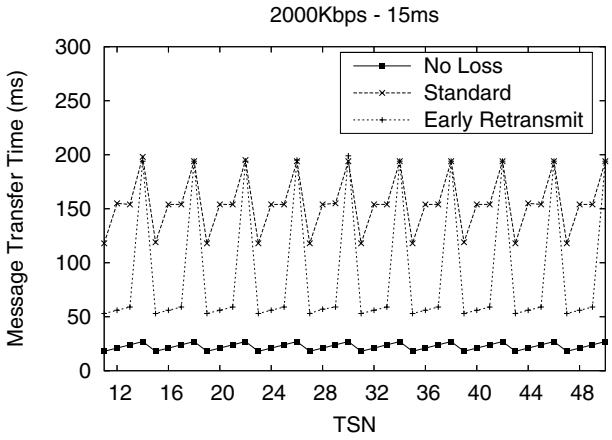

Figure 2. MTT of individual messages, using bursts of four messages and 200 ms inter-burst gaps

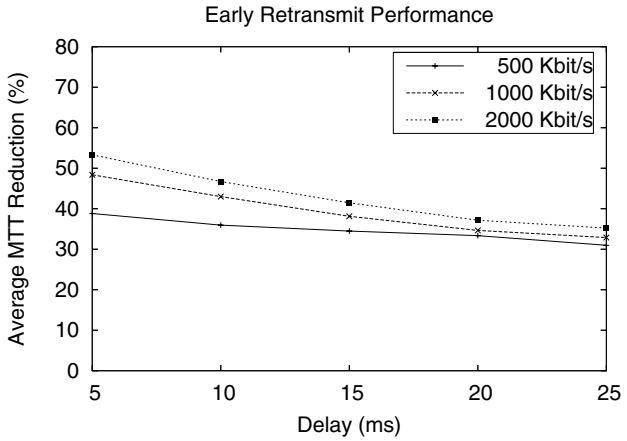

Figure 3. Average MTT reduction achieved by Early Retransmit, using bursts of four messages, and 200 ms inter-burst gaps

sages, when no packet loss occurred. The reason to why the MTT increased linearly within the bursts is the queuing that occurred at the bottleneck. When messages were lost within the emulated network we can see large differences in their MTT, depending on which SCTP version that was used. For the unmodified SCTP implementation, labeled "Standard", we can see that the time required to transfer a message was in the range $110 - 200$ ms. The Early Retransmit version only required about 60 ms, with an exception for the last message in each burst that, similar to standard SCTP, required about 200 ms.

The largest performance improvement can be found when the message with TSN 12 was lost. The MTT reduction, provided by Early Retransmit, for a loss of this message was approximately 64%. Averaged over all loss positions in this scenario, Early Retransmit was able to reduce the MTT with 41%. The reason to the performance enhancement, given by Early Retransmit, is simply that the dynamic reduction of the duplicate acknowledgment threshold permits fast retransmit of lost packets, which the standard SCTP version is unable to offer[4].

In addition to this, the results also reveal a performance problem associated with the management of the retransmission timer. As can be seen in Figure 2, for the standard SCTP implementation there were rather large variations in the MTT within the bursts. For example, the first message in each burst required about 110 ms, and the fourth about 200 ms. The reason for this difference is that the retransmission timer is restarted each time an acknowledgment, which cumulatively acknowledges new data, is received (as defined in the SCTP specification [19]). For example, if the first message in a burst is lost, the timer will not be restarted, because no acknowledgment will advance the cumulative acknowledgment level. Thus, the first message will be retransmitted according to the current value of the retransmission timer. However, if the second message is lost, the retransmission timer will be restarted when the acknowledgment that, cumulatively, acknowledges the first message arrives. Thus, the timeout will occur one round-trip time later, even if the value of the timer is the same. This phenomenon has been observed, and discussed, for TCP as well [6][10].

Figure 3 summarizes all results achieved when using a burst size of four messages, and 200 ms inter-burst gaps. The *y*-axis of this graph shows the average reduction in MTT, provided by Early Retransmit. The *x*-axis of the graph shows the different end-to-end delays, and the different lines in the graph represent the different bandwidths evaluated. The largest performance improvement can be found when the bottleneck bandwidth was high (2000 Kbit/s) and the one way end-to-end delay was low (5 ms). For this combination, Early Retransmit reduced the MTT with approximately 53%, averaged over all loss positions. For the combination of low bandwidth (500 Kbit/s), and high delay (25 ms), the average performance gain was less, but still significant. Here, Early Retransmit was able to reduce the MTT with 31%. This difference is heavily dependent on the minimum allowed value of the retransmission timer. As the end-to-end delays were relatively small, and stable, in all the experiments conducted, the RTO calculation at the sender resulted in $\texttt{RTO}_{\texttt{min}}$ (100 ms) all the time. Thus, regardless of bottleneck bandwidth and end-to-end delay, loss detection by retransmission timeout required the same amount of time. Early Retransmit, how-

[4]The SCTP implementation included in version 2.6.15 of the Linux kernel uses a duplicate acknowledgment threshold of four, in accordance with the SCTP specification [19]. However, in the SCTP errata document [18] the threshold has been reduced to three. Thus, in later versions of this implementation, it could be expected that the threshold is reduced, and that fast retransmit is possible for the first message in each burst.

 ISBN # 1-56555-317-9

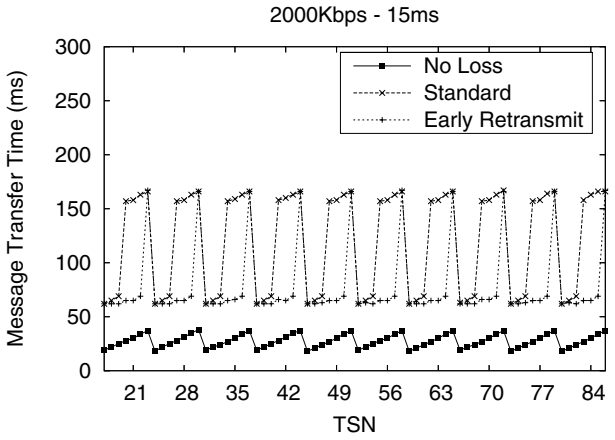

Figure 4. MTT of individual messages, using bursts of seven messages and 100 ms inter-burst gaps

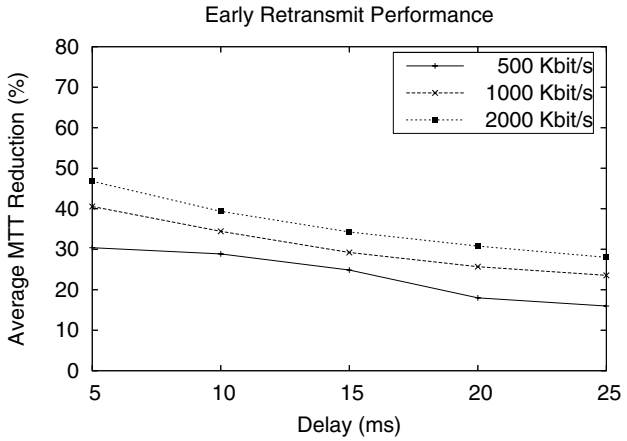

Figure 5. Average MTT reduction achieved by Early Retransmit, using bursts of seven messages, and 100 ms inter-burst gaps

ever, is not limited by any other factor than the total network delay. Thus, as long as the bandwidth was increased, and the end-to-end delay was reduced, Early Retransmit detected loss quicker.

In Figure 4 another example of the results is shown. Like in the previous results, the bottleneck bandwidth was 2000 Kbit/s and the delay 15 ms. However, for these experiments, the size of the bursts was somewhat larger, seven messages, and the inter-burst gaps shorter, 100 ms. From this graph we can see that the performance enhancement given by Early Retransmit was significant. The largest improvement, provided by Early Retransmit, can be found when the message with TSN 84 was lost. The MTT reduction for this message was 60%. Averaged over all loss positions for this scenario, the MTT reduction was 34%. However, the main reason for the performance gain was not the same. In the previous results, the performance difference was due to the fact that the standard version of SCTP was not able to trigger fast retransmit at all, and thus needed to rely on retransmission timeouts. In this case, however, the standard version also allowed fast retransmit for all losses. For the first three messages in each burst this is clearly visible, as their MTTs are comparable to those of Early Retransmit. For the last four messages in each burst this is not so obvious, as the Early Retransmit version outperformed the standard version. The performance difference depends on the number of duplicate acknowledgments that are needed for fast retransmit, and the length of the inter-burst gaps. As the standard version requires four duplicate acknowledgments, only the first three messages in each burst can be fast retransmitted immediately. The last four can not be retransmitted until duplicate acknowledgments generated by the next burst arrives. The reason why fast retransmit was invoked for a loss of these messages, instead of retransmission due to timeout, is simply that the time needed for the retransmission timer to expire was longer than the inter-burst gaps. The Early Retransmit version, however, does not have to wait for duplicate acknowledgments generated by the next burst, as it dynamically lowers its threshold when the amount of outstanding data gets below four packets.

Figure 5 summarizes all results achieved when using a burst size of seven messages, and 100 ms inter-burst gaps. The *y*-axis of the graph shows the average reduction in MTT, when using Early Retransmit. Further, the *x*-axis shows the different end-to-end delays used, and the different lines represent different bandwidths evaluated. The largest performance improvement can be found when the bottleneck bandwidth was high (2000 Kbit/s) and the one way end-to-end delay was low (5 ms). For this combination, Early Retransmit was able to reduce the MTT with approximately 47%. For the combination of low bandwidth (500 Kbit/s), and high delay (25 ms), the average performance gain was less significant. Here, the average MTT reduction, provided by Early Retransmit, was 16%. The reason for this difference is similar to that of the previously mentioned results, those with smaller burst size. However, for these experiments (with larger bursts) the loss detection of the standard version was not limited by the $\texttt{RTO}_{\texttt{min}}$ value. Instead, the time needed for loss detection was affected by the inter-burst gaps, which limited how fast the duplicate acknowledgments could return from the receiver.

Furthermore, by having a larger number of messages in each burst, together with smaller inter-burst gaps, the performance benefit of using Early Retransmit was generally decreased. The reason for this is simply that the standard version of SCTP permitted fast retransmit of all lost messages, and that it could perform it equally fast for the three first messages in each burst. The performance benefit that Early Retransmit gave for these experiments was instead due to the

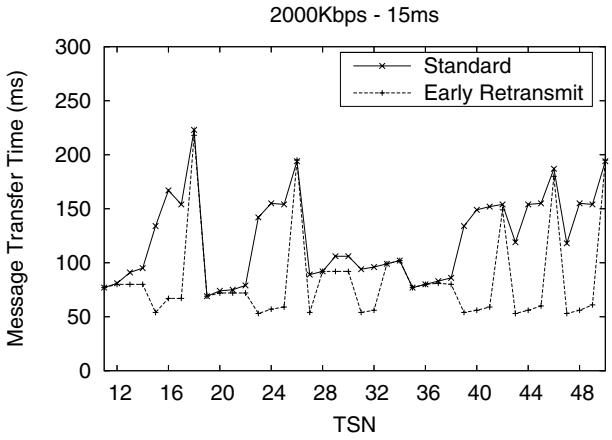

Figure 6. MTT of individual messages, using bursts of four messages and exponentially distributed inter-burst gaps with mean 200 ms

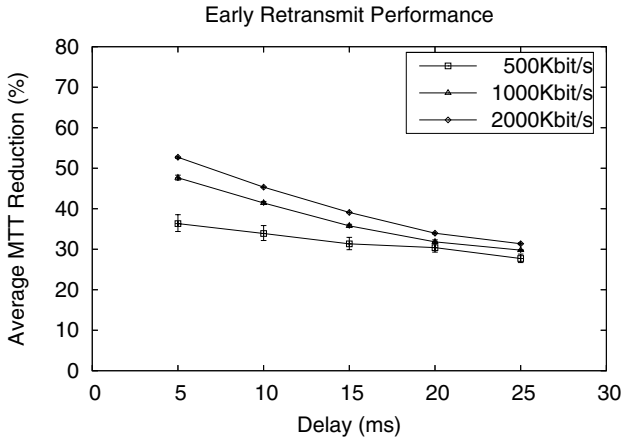

Figure 7. Average MTT reduction achieved by Early Retransmit, using bursts of four messages, and exponentially distributed inter-burst gaps with mean 200 ms

smaller amount of duplicate acknowledgments required for loss detection of messages at the end of bursts.

To summarize the results for the experiments with static burst sizes and static inter-burst gaps, we can conclude that Early Retransmit is able to provide significant improvements in timely loss recovery. As could be expected, the benefit of using Early Retransmit was largest when small bursts, separated by long inter-burst gaps, were transmitted.

4.2. Static Burst Sizes & Random Inter-burst Gaps

In Figure 6, an example of the results for static burst sizes and random inter-burst gaps is shown. The *y*-axis shows the MTT of individual SCTP messages, and the *x*-axis shows the TSN of the corresponding message. For the experiments shown in this graph ten bursts, each containing four messages, were measured. Furthermore, the bottleneck bandwidth was 2000 Kbit/s, the delay 15 ms, and the time between the bursts exponentially distributed with a mean of 200 ms. As could be expected, it is now much harder to identify different bursts by a general pattern in the MTT. However, it is clearly visible that Early Retransmit offers a general performance enhancement. The largest improvement can be found when the message with TSN 48 was lost. For a loss of this message, Early Retransmit reduced the MTT with 64%. The average reduction in MTT for this particular scenario was 59.5%.

Figure 7 contains the average MTT reduction that Early Retransmit provided, when 40 replications were made for each loss position. In this graph we can see the average reduction in MTT for all combinations of bottleneck bandwidth and end-to-end delay, when the burst size was four messages and the inter-burst gaps were exponentially distributed with a mean of 200 ms. Further, this graph also provides 95% confi-

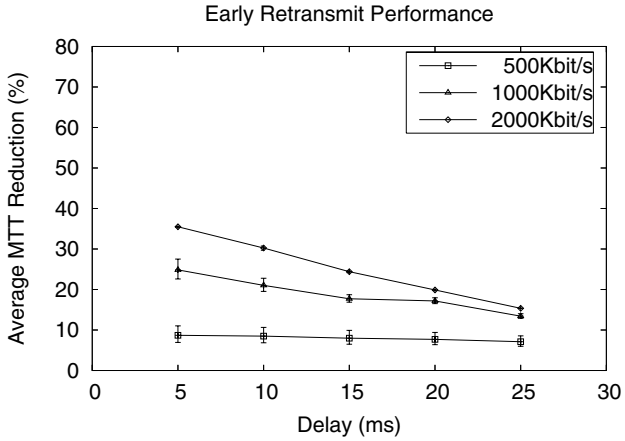

Figure 8. Average MTT reduction achieved by Early Retransmit, using bursts of seven messages, and exponentially distributed inter-burst gaps with mean 100 ms

dence intervals for the different combinations. As can be seen in the graph, the benefit of using Early Retransmit instead of the standard SCTP loss recovery was significant, and very stable. In fact, the confidence intervals for the experiments with a bandwidth greater or equal to 1000 Kbit/s are barely visible. Consistent with previous results, the highest performance improvement was achieved when the end-to-end delay was low, and the bandwidth high. The overall improvement is still large for all combinations evaluated. If we compare these results with the results obtained for the static counterpart, shown in Figure 3, we can see that the performance improvement is almost the same. For these experiments, the MTT reduction was between 53% and 28%, and in the static scenario between 53% and 31%.

For the scenario with larger bursts and shorter inter-burst

gaps, shown in Figure 8, the performance improvement was slightly less. This was expected, and consistent with the results for the static experiments shown in Figure 4, as fast retransmit was invoked more often by the standard SCTP implementation. However, the improvement provided by Early Retransmit for these experiments was significantly lower than for the static experiments. Especially for a bandwidth of 500 Kbit/s, which had an almost constant MTT reduction of less than 10%. The reason to why these results differ so much from the static experiments can be explained by considering the properties of exponential distributions. Consider an exponential distribution with rate parameter λ. Using the cumulative distribution function of exponential distributions, shown in Equation 1, it can be concluded that a majority of all random variables drawn from this distribution will be less than the mean $(1/\lambda)$. This because $F(1/\lambda;\lambda) = 1 - e^{-\lambda * 1/\lambda} = 0.632\ldots \approx 0.63$. Thus, approximately 63% of all random variables $X \sim Exponential(\lambda)$ will result in a $X \leq 1/\lambda$.

$$F(x;\lambda) = \begin{cases} 1 - e^{-\lambda x} & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases} \quad (1)$$

Therefore, a majority of all inter-burst gaps was shorter than the mean value, which already was low for these experiments, causing some parts of the traffic to be almost continuous. Thus, the improvement given by Early Retransmit was less significant, as there in many cases were enough packets in flight for the standard SCTP implementation to invoke fast retransmit as well. In addition, the few long gaps that were generated by the exponential distribution did not affect the performance improvement of Early Retransmit so much either. The reason for this is that the performance enhancement provided by Early Retransmit is not proportional to the length of the inter-burst gaps. That is, if a packet loss occurs and the gap is long enough to cause a retransmission timeout, then the actual length of the gap will not affect loss recovery time in any way.

4.3. Random Burst Sizes & Static Inter-burst Gaps

In Figure 9, an example of the results for random burst sizes with static inter-burst gaps is shown. The size of each burst was drawn from a discrete uniform distribution with range $[1,7]$, and the inter-burst gaps were all 200 ms. Like in all previous examples, the bottleneck bandwidth was 2000 Kbit/s and the end-to-end delay 15 ms. Consistent with the previous results, it is easy to see that Early Retransmit provided considerable performance enhancements in timely loss recovery. For this particular loss scenario the MTT was reduced with as much as 67%, when the message with TSN 49 was lost. Averaged over all loss positions in this scenario, Early Retransmit was able to reduce the MTT with 36%.

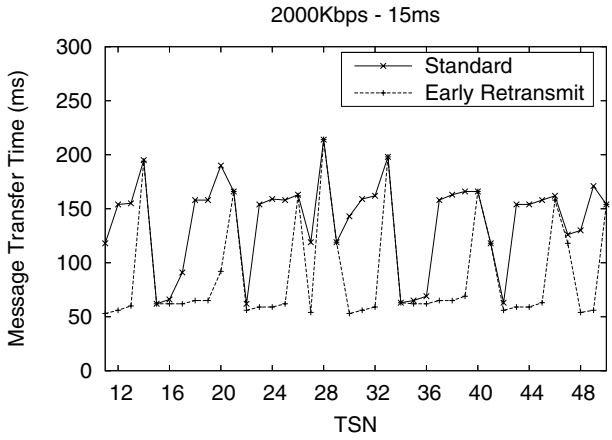

Figure 9. MTT of individual messages, using uniformly distributed burst with a mean of four, and 200 ms inter-burst gaps

Figure 10 shows the average MTT reduction for all combinations of bandwidth and end-to-end delay. This graph shows an average of 40 replications, for every combination of bandwidth, end-to-end delay and loss position, together with 95% confidence intervals. As can be seen in the graph, Early Retransmit provided significant reductions in average MTT, for all combinations of bandwidth and end-to-end delay. Compared to the static counterpart, shown in Figure 3, we can see that the average reduction in MTT was somewhat smaller, ranging from 29% to 46%. The large confidence intervals indicate that the MTT reduction was highly variable. The large variations are however natural, as the possible range of burst sizes lies in a region where the benefit of Early Retransmit varies considerably. For instance, if bursts are of size one neither Early Retransmit nor the standard implementation can invoke fast retransmit. Thus, Early Retransmit will not improve the loss recovery performance. If bursts are larger, then it may be possible for both Early Retransmit and the standard implementation to trigger fast retransmit, depending on which packets that are lost and how large the bursts are.

The results from the experiments with burst sizes drawn from a discrete uniform distribution with range $[4,10]$, and static inter-burst gaps of 100 ms, are shown in Figure 11. This graph shows the average reduction in MTT of 40 replications for every loss position, with 95% confidence intervals. The average improvement in timely loss recovery is very similar to the static scenario, shown in Figure 5. The reason why these results are more similar to the static scenario, compared to the previously mentioned results, is that the range of possible burst sizes lies in a range where the benefit of Early Retransmit is less variable. This also makes the results less prone to variations, as indicated by the relatively small confidence intervals.

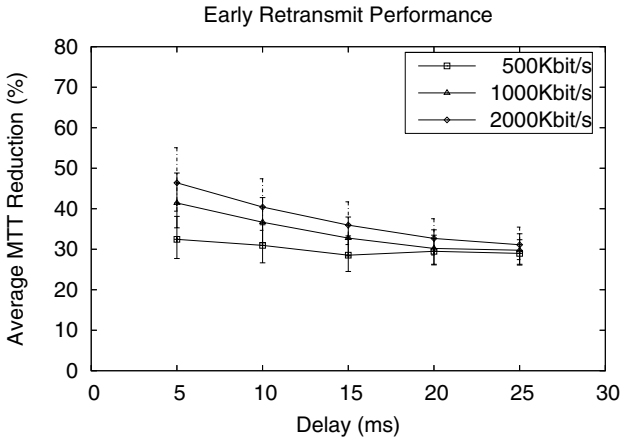

Figure 10. Average MTT reduction achieved by Early Retransmit, using randomized burst sizes with a mean of four, and 200 ms inter-burst gaps

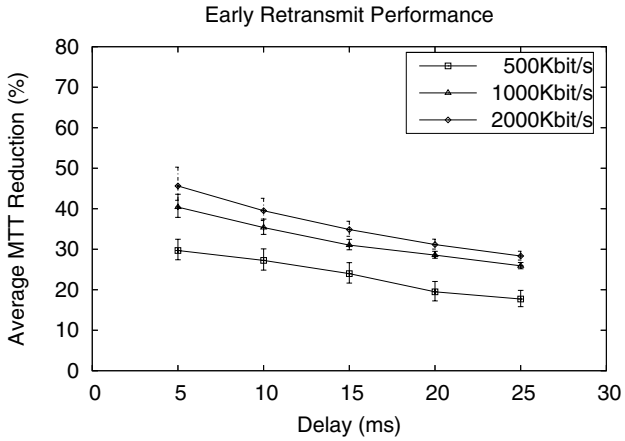

Figure 11. Average MTT reduction achieved by Early Retransmit, using randomized burst sizes with a mean of seven, and 100 ms inter-burst gaps

5. CONCLUSIONS

This paper studies the SCTP loss recovery mechanisms, and their inefficiency, when the amount of outstanding data is small. By comparing some proposals that try to enhance the SCTP loss recovery in such situations, we conclude that the Early Retransmit mechanism is the most appropriate for signaling traffic.

However, as Early Retransmit was designed for both TCP and SCTP, the original specification is not adapted appropriately to non-bulk traffic, such as signaling traffic. Therefore, we propose a non-bulk adaptation to Early Retransmit, which we also evaluate in a signaling context.

Using an emulated network environment, we show that Early Retransmit is able to reduce the time needed for loss recovery, in a standard SCTP implementation (lksctp), significantly. In some cases the time needed for transmission, and retransmission, of lost packets was reduced with as much as 67%. The results show that the largest performance improvements, provided by Early Retransmit, can be achieved when small bursts of packets, separated by long idle periods, are transmitted. When transmitting traffic that consists of larger bursts, and shorter idle periods, the performance improvement is less but still significant.

The results also show that the management of the SCTP retransmission timer, alone, can cause loss recovery time to almost double in some situations. For future work we intend to evaluate the impact of this phenomenon more closely.

REFERENCES

[1] M. Allman, K. Avrachenkov, U. Ayesta, and J. Blanton. Early Retransmit for TCP and SCTP. Internet draft, Internet Engineering Task Force, November 2006.

[2] M. Allman, H. Balakrishnan, and S. Floyd. Enhancing TCP's Loss Recovery Using Limited Transmit. RFC 3042, Internet Engineering Task Force, January 2001.

[3] A. T. Andersen. *Modelling of packet traffic with matrix analytic methods*. PhD thesis, Technical University of Denmark, DTU, 1995.

[4] E. Blanton and M. Allman. Using TCP Duplicate Selective Acknowledgment (DSACKs) and Stream Control Transmission Protocol (SCTP) Duplicate Transmission Sequence Numbers (TSNs) to Detect Spurious Retransmissions. RFC 3708, Internet Engineering Task Force, February 2004.

[5] IBM Corporation. The lksctp Project. http://lksctp.sourceforge.net.

[6] H. Ekström and R. Ludwig. The peak-hopper: A new end-to-end retransmission timer for reliable unicast transport. In *Proceedings of the IEEE INFOCOM*, 2004.

[7] J. Garcia, S. Alfredsson, and A. Brunstrom. The impact of loss generation on emulation-based protocol evaluation. In *Proceedings of the International Conference on Parallel and Distributed Computing and Networks (PDCN 2006)*, February 2006.

[8] L. Gharai, C. Perkins, and T. Lehman. Packet reordering, high speed networks and transport protocol performance. In *Proceedings of the International Conference on Computer Communications and Networks (ICCCN 2004)*, October 2004.

[9] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Measurement and classification of out-of-

sequence packets in a tier-1 IP backbone. In *Proceedings of the IEEE INFOCOM*, 2003.

[10] R. Ludwig and K. Sklower. The eifel retransmission timer. *SIGCOMM CCR*, 30(3):17–27, 2000.

[11] M. Mellia, M. Meo, and C. Casetti. TCP Smart Framing: A Segmentation Algorithm to Reduce TCP Latency. *IEEE/ACM TON*, 13(2):316–329, April 2005.

[12] L. Ong, I. Rytina, M. Garcia, H. Schwarzbauer, L. Coene, H. Lin, I. Juhasz, M. Holdrege, and C. Sharp. Framework architecture for signaling transport. RFC 2719, Internet Engineering Task Force, October 1999.

[13] J. Postel. Transmission Control Protocol. RFC 793, Internet Engineering Task Force, September 1981.

[14] L. Rizzo. Dummynet: A Simple Approach to the Evaluation of Network Protocols. *ACM CCR*, 27(1):31–41, 1997.

[15] J. Rosenberg, H. Schulzrinne, and G. Camarillo. The stream control transmission protocol (SCTP) as a transport for the session initiation protocol (SIP). RFC 4168, Internet Engineering Task Force, October 2005.

[16] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, Internet Engineering Task Force, June 2002.

[17] F. J. Scholtz. Statistical analysis of common channel signaling system no. 7 traffic. In *Proceedings of the 15th Internet Traffic Engineering and Traffic Management (ITC) Specialist Seminar*, Wurzburg, Germany, 2002.

[18] R. Stewart, I. Arias-Rodriguez, K. Poon, A. Caro, and M. Tuexen. Stream Control Transmission Protocol (SCTP) Specification Errata and Issues. RFC 4460, Internet Engineering Task Force, April 2006.

[19] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. RFC 2960, Internet Engineering Task Force, October 2000.