Computer Communications 33 (2010) 1982–1991

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Throughput analysis of Non-Renegable Selective Acknowledgments (NR-SACKs) for SCTP ☆

Ertugrul Yilmaz [a], Nasif Ekiz [a], Preethi Natarajan [b], Paul D. Amer [a,*], Jonathan T. Leighton [a], Fred Baker [c], Randall R. Stewart [d]

[a] CIS Dept., University of Delaware, United States
[b] Cisco Systems, 425 East Tasman Drive, San Jose, CA 95134, USA
[c] Cisco Systems, 1121 Via Del Rey, Santa Barbara, CA 93117 USA
[d] Huawei Technologies, Chapin, SC 29036, USA

ARTICLE INFO

Article history:
Received 10 December 2009
Received in revised form 29 June 2010
Accepted 30 June 2010
Available online 27 July 2010

Keywords:
CMT
TCP
SACK
SCTP
Reneging

ABSTRACT

Preliminary work introduced Non-Renegable Selective Acknowledgments (NR-SACKs) and showed they (i) better utilize a data sender's memory in both SCTP and CMT, and (ii) improve throughput in CMT. In this paper, we provide the latest specification of NR-SACKs, and extend the investigation of throughput improvements that NR-SACKs can provide. Using ns-2 simulation, for various loss conditions and bandwidth-delay combinations, we show that the throughput observed with NR-SACKs is at least equal and sometimes better than the throughput observed with SACKs. We introduce "region of gain" which defines for a given bandwidth, delay, and send buffer size combination, what range of loss rates results in significant throughput improvement when NR-SACKs are used instead of SACKs. In both SCTP and CMT, NR-SACKs provide greater throughput improvement as the send buffer size decreases, and as end-to-end delay decreases. Provided that the bandwidth-delay product (BDP) $\geqslant$ send buffer size, additional bandwidth does not increase NR-SACKs' throughput improvements for either SCTP or CMT. For BDPs < send buffer size, the throughput improvement decreases as the BDP decreases.

1. Introduction

Both TCP and SCTP employ selective acknowledgements (SACKs) that provide a data sender with a more accurate view of the data received at the data receiver than is provided by only using cumulative acknowledgments (cum-acks). Although SACKs inform a data sender of the reception of out-of-order data, RFC2018 [10] and RFC4960 [11] specify SACKs as being advisory in TCP and SCTP, respectively. A data receiver is permitted to later discard data that it previously selectively-acked (herein, "SACKed") without having delivered the data to the receiving application. This "SACK and discard without delivery" behavior is called *reneging*. In practice, reneging should only occur during drastic situations, such as when an OS needs to reclaim previously allocated buffer space to keep a system from crashing.

Due to the *potential* need to retransmit in case reneging occurs, with the current SACK mechanism a data sender is not released from the responsibility of retransmitting data until it is cum-acked. Hence a copy of all SACKed data must be kept in the send buffer until that SACKed data also is cum-acked.

In SCTP, situations commonly exist where out-of-order data are non-renegable, that is, the data receiver cannot renege. For example, in addition, SCTP multistreaming allows data transfer over multiple independent logical streams. It is common SCTP behavior for data received in-order within a stream to be delivered to the application even if the data is out-of-order with respect to the overall flow.

SCTP defines an unordered data transfer service. Data marked for unordered delivery can be delivered to the application immediately upon arrival, regardless of the data's position in the overall flow. Once data has been delivered, by definition, it no longer can be reneged.

In this paper, we discuss Non-Renegable Selective Acknowledgements (NR-SACKs), a new ack mechanism that enables a transport receiver to explicitly identify that some or all out-of-order data

☆ Prepared through collaborative participation in the Communication and Networks Consortium sponsored by the US Army Research Lab under Collaborative Tech Alliance Program, Coop Agreement DAAD19-01-2-0011. The US Govt. is authorized to reproduce and distribute reprints for Govt. purposes notwithstanding any copyright notation thereon. Supported by the University Research Program, Cisco Systems, Inc.

* Corresponding author. Tel.: +1 302 831 1944; fax: +1 302 831 8458.
E-mail addresses: eyilmaz@cis.udel.edu (E. Yilmaz), nekiz@cis.udel.edu (N. Ekiz), prenatar@cisco.com (P. Natarajan), amer@cis.udel.edu (P.D. Amer), leighton@cis.udel.edu (J.T. Leighton), fred@cisco.com (F. Baker), randall@lakerest.net (R.R. Stewart).

0140-3664/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2010.06.028

are non-renegable [1,2]. This information about the "renegability" of received out-of-order data is in addition to all information carried by conventional SACKs. Preliminary work defined an early version of NR-SACKs, and investigated their performance benefits for unordered data transfers in SCTP and CMT, a variation of SCTP that permits concurrent transfer over multiple paths [6]. The results showed that NR-SACKs improve *memory utilization* both in CMT and SCTP, and improve throughput in CMT. The loss conditions and bandwidth-delay combination initially studied demonstrated limited throughput improvement in SCTP.

In this work we first present in Section 2 an example that demonstrates the limitations of SACKs. Section 3 then presents a stable version of NR-SACKs developed as a result of feedback from several IETF presentations. The NR-SACK semantics as well as format significantly changed since first presented in [1]. In Sections 4 and 5, we respectively present an ns-2 simulation model and investigation that extends beyond [1] by focusing on the throughput gains for SCTP and CMT data transfers using SACKs vs. NR-SACKs for a broader range of loss rate, bandwidth and delay combinations.

Specifically, we investigate how send buffer size, loss rate, bandwidth or delay alone affects the throughput improvement that NR-SACKs provide when the other parameters are constant. In addition, we investigate how throughput improvement is affected when the bandwidth-delay product (BDP) is equal to, smaller than, and larger than the send buffer size. In Section 6, we discuss the management of how NR-SACKs are negotiated by peer endpoints. Finally Section 7 provides some concluding remarks.

2. Motivation: selective acks can be inefficient

2.1. Transport layer send buffer

As shown in Fig. 1, data in a reliable transport's (e.g., SCTP, TCP) send buffer can be classified as either: new application data waiting to be transmitted for the first time, or copies of data that have been transmitted, and are yet to be cum-acked, a.k.a. the *retransmission queue* (RtxQ).

The transport data sender is responsible for the data in the RtxQ until informed by the receiver that either it (1) has delivered the data, or (2) guarantees to eventually deliver the data to the receiving application. In traditional in-order data delivery service, (1) or (2) is achieved when a receiver cum-acks the latest in-order data. Cum-acked data either has been delivered to the application or is deliverable (ready for delivery). In either case, cum-acks are an explicit assurance from the receiver not to renege on the cum-acked data.

On receiving a cum-ack, a sender is no longer responsible for the cum-acked data, and removes the corresponding data from the RtxQ. In the current SCTP (and TCP) specifications, a data receiver can inform a data sender of non-renegable data *only* with cum-acks. All SACKed out-of-order data is renegable, and might need to be retransmitted.

As explained in Section 1, SCTP's multistreaming and unordered data delivery services result in situations where out-of-order data is delivered even before the data is cum-acked, and is thus non-renegable.

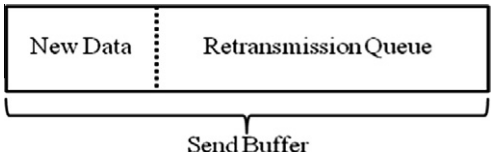

Fig. 1. Transport layer send buffer.

Furthermore, some operating systems allow configuration of transport layer implementations such that no out-of-order data (deliverable or not deliverable) can be reneged. For example, in FreeBSD, the sysctl parameters net.inet.tcp.do_tcpdrain and net.inet.sctp.do_sctp_drain can be configured such that the OS never reclaims buffer space previously allocated to TCP and SCTP sockets, respectively [7]. When the drain option is off at the data receiver, any SACKed out-of-order data is guaranteed never to be reneged. SACKs cannot relay information about renegability of out-of-order data. NR-SACKs allow a data receiver to inform a data sender of non-renegable out-of-order data so that all unnecessary copies of non-renegable data in the sender's RtxQ can be removed sooner than if traditional SACKs were used, resulting in better utilization of the memory allocated for the send buffer, and in some cases, improved throughput.

To better understand NR-SACKs, we present an example reliable data transfer first using SACKs, and then in Section 3.2 using NR-SACKs.

2.2. SCTP unordered data transfer using SACKs

While out-of-order data can become non-renegable in several ways, this discussion uses a simple unordered data transfer example, shown in Fig. 2. The SCTP send buffer, denoted by the box on the left, holds a maximum of eight TPDUs. Each SCTP PDU is assigned a unique *Transmission Sequence Number (TSN)*. The time slice shown in Fig. 2 picks up the data transfer at a point when the sender's congestion window (cwnd) $C = 8$, allows transmission of 8 TPDUs (arbitrarily numbered TSNs 11–18). Once TSN 18 is transmitted, the RtxQ has grown to fill the entire send buffer.

In Fig. 2's scenario, TSN 11 is presumed lost in the network. TSNs 12–18 are received out-of-order and immediately delivered to the application, and SACKed by the SCTP receiver. SACKs are formatted as:

S : CumAckTSN; GapAckStart-GapAckEnd

The GapAckStart and GapAckEnd values are relative to the cum-ack value, and together they specify a block of TSNs received out-of-order. At the sender, the first received SACK (S:10;2–2) is a dupack, and gap-acks TSN 12. Though data corresponding to TSN 12 has been delivered to the receiving application, this SACK does not (or more precisely, cannot) convey the non-renegable nature of TSN 12, forcing the sender to keep TSN 12 in the RtxQ. Starting from this point, the buffering of TSN 12 is unnecessary; it wastes send buffer space.

The gap-ack for TSN 12 reduces the amount of outstanding data (O) to 7 TPDUs. Since $O < C$, the sender could in theory transmit new data, but in practice cannot do so since the completely filled send buffer *blocks* the sending application from writing new data into the transport layer. We call this situation *send buffer blocking*. Note that send buffer blocking prevents the sender from fully utilizing its cwnd.

Similarly, after receiving (S:10;2–3) and (S:10;2–4), the sender needlessly maintains copies of TSNs 13 and 14, respectively. These copies needlessly use kernel memory, and send buffer blocking continues to prevent new data transmission. On receipt of (S:10;2–4), i.e., the third dupack, the sender halves the cwnd ($C = 4$), fast retransmits TSN 11, and enters fast recovery. Dupacks received during fast recovery further increase the amount of unnecessary data in the RtxQ, prolonging inefficient RtxQ usage. Note that even though these dupacks reduce outstanding data ($O < C$), send buffer blocking prevents new data transmission.

The sender eventually exits fast recovery when the SACK for TSN 11's retransmission (S:18) arrives. The sender removes TSNs 12–18 from the RtxQ, and concludes the current instance of send buffer blocking. Since send buffer blocking prevented the sender

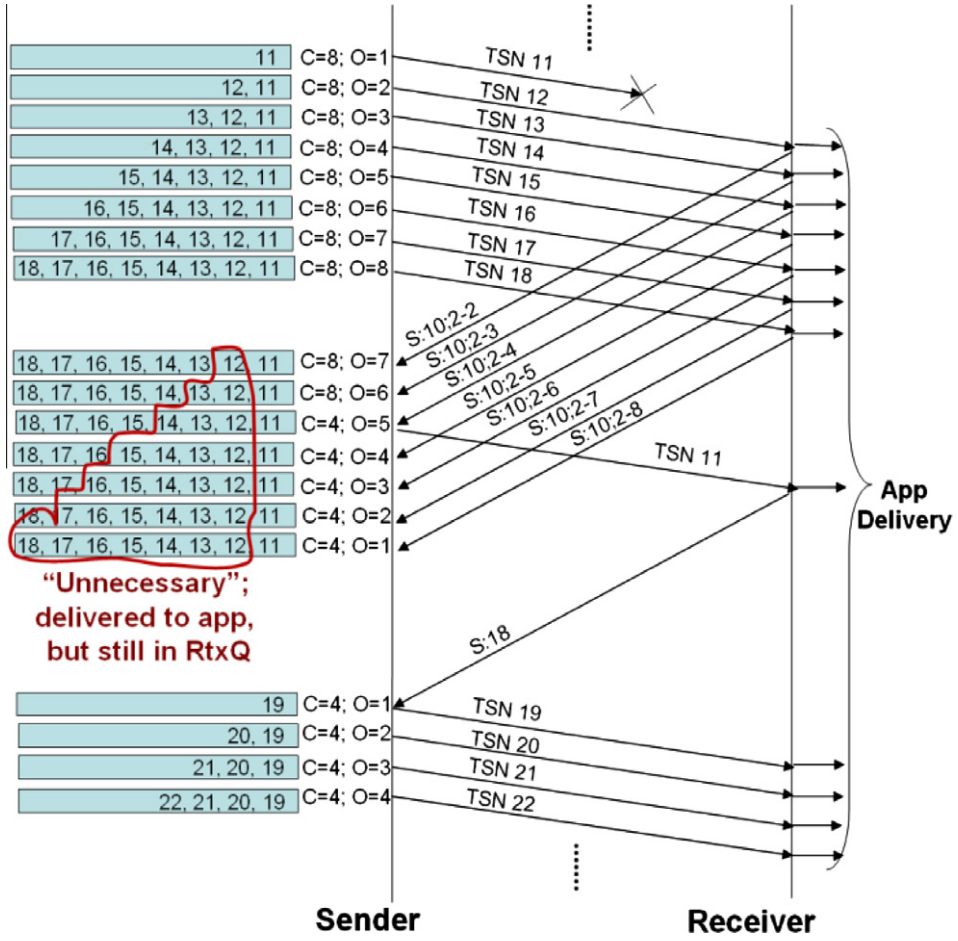

Fig. 2. Unordered SCTP data transfer using SACKs.

from fully utilizing the cwnd before, the new cum-ack (S:18) does not increase the cwnd [11]. The sending application writes data into the newly available send buffer space, and the sender now transmits TSNs 19–22.

Based on Fig. 2's timeline, the following observations can be made regarding transfers with non-renegable out-of-order data:

- The copies of delivered out-of-order data unnecessarily occupy kernel memory (RtxQ). The amount of wasted memory is a function of *flightsize* (amount of data "in flight") during a loss event; a larger *flightsize* wastes more memory.
- When the RtxQ grows to fill the entire send buffer, send buffer blocking ensues, which degrades throughput.

2.3. CMT using SACKs

SCTP supports *transport layer multihoming* for fault-tolerance purposes [11]. An SCTP association binds multiple IP addresses at each endpoint, but chooses one destination address as *primary*, which is used for all data traffic under normal transmission. Failure in reaching the primary destination results in failover, where an SCTP endpoint dynamically chooses an alternate destination to transmit data.

Concurrent Multipath Transfer (CMT) is an experimental SCTP extension that further exploits multihoming beyond fault tolerance for simultaneous data transfer over multiple independent paths [6]. Similar to an SCTP sender, the CMT sender uses a single send buffer and RtxQ for data transfer. However, the CMT sender's total *flightsize* is the sum of *flightsizes* across all paths. Since the amount of potential unnecessary kernel memory usage and the probability of send buffer blocking increase as a transport sender's flightsize increases, we hypothesize that a CMT association is even more likely than an SCTP association to suffer from the inefficiencies of the existing SACK mechanism.

Wischik et al. [9] discuss the advantages of resource pooling and argues the need for multipath-capable TCP (a.k.a. multipath TCP) to facilitate resource pooling. Similar to CMT, multipath TCP extends TCP to support multihoming, and sets up multiple subflows to simultaneously transmit data over multiple paths. Similar to a CMT sender, a multipath TCP sender's flightsize would be the sum of flightsizes across all paths. Therefore, we expect that our hypothesis for CMT and the corresponding observations would be pertinent to other multipath transports such as multipath TCP.

3. Non-Renegable Selective Acknowledgments

Non-Renegable Selective Acknowledgments (NR-SACKs) enable a receiver to explicitly convey the renegable vs. non-renegable nature of out-of-order data. NR-SACKs provide the same information as SACKs for congestion and flow control, and the sender is expected to process this information identical to SACK processing. In addition, NR-SACKs provide the added option to report some or all of the out-of-order data as being non-renegable.

3.1. NR-SACK details

The proposed NR-SACK for SCTP is shown in Fig. 3. Since NR-SACKs extend SACK functionality, an NR-SACK has several

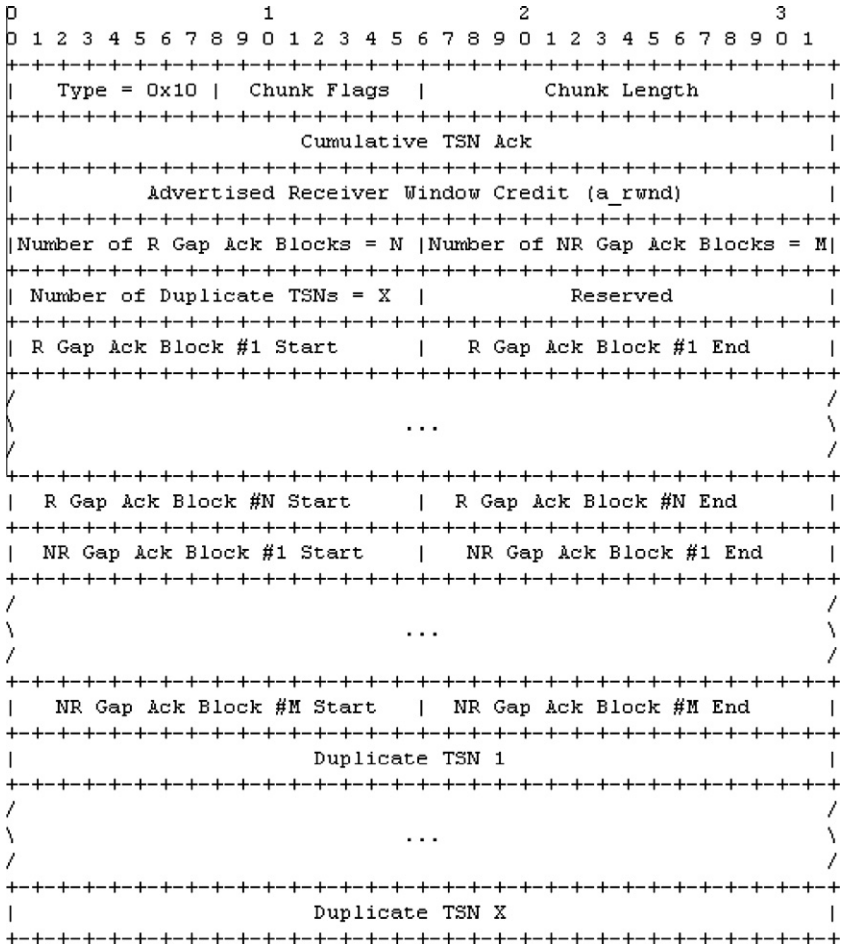

Fig. 3. NR-SACK for SCTP.

fields identical to a SACK: the *Cumulative TSN Ack*, the *Advertised Receiver Window Credit*, and *Duplicate TSNs*. These fields have identical semantics to the corresponding fields in the SACK [11]. Further, an NR-SACK's *R Gap Ack Blocks* are semantically equivalent to a SACK's *Gap Ack Blocks*.

An NR-SACK also contains *NR Gap Ack Blocks*, a.k.a. "nr-gap-acks". Each NR Gap Ack Block acknowledges a continuous subsequence of non-renegable out-of-order data. All data with TSNs $\geqslant$ (Cumulative TSN Ack + NR Gap Ack Block Start) and $\leqslant$ (Cumulative TSN Ack + NR Gap Ack Block End) of each NR Gap Ack Block are reported as both *gap-acked* and *non-renegable*. The *Number of NR Gap Ack Blocks (M)* field indicates the number of NR Gap Ack Blocks included in the NR-SACK.

Note that TSNs listed in both R Gap Ack and NR Gap Ack Blocks are being gap-acked. While TSNs listed in R Gap Ack Blocks may be reneged, TSNs listed in NR Gap Ack Blocks are non-renegable. These two lists should be disjoint.

Importantly, non-renegable information cannot be revoked. If a TSN is nr-gap-acked in an NR-SACK, then all subsequent NR-SACKs must also nr-gap-ack that TSN. Complete details of the NR-SACK can be found in [4].

Each NR-SACK has a constant overhead of 4 bytes: 2 bytes for the "Number of NR Gap Ack Blocks" field, and 2 bytes for "RESERVED" padding to align Gap Ack Blocks on a 4-byte word boundary. No additional overhead occurs if all data is exclusively renegable or non-renegable. In the worst case, for each instance when an NR Gap Ack block resides in the middle of what would be a single gap-ack block within a SACK, an NR-SACK would need 8 extra bytes.

3.2. SCTP unordered data transfer using NR-SACKs

With NR-SACKs, an SCTP receiver can optionally convey the non-renegable nature of out-of-order TSNs. The sender no longer needs to keep nr-gap-acked TSNs in the RtxQ, thus allowing the sender to free up kernel memory sooner than if the TSNs were only gap-acked.

Fig. 4 is analogous to Fig. 2's example; this time the data transfer employs NR-SACKs instead of SACKs. The sender and receiver are assumed to have negotiated the use of NR-SACKs during association establishment (discussed in Section 6). As in Fig. 2, TSNs 11–18 are initially transmitted, and TSN 11 is presumed lost. For each TSN arriving out-of-order, the SCTP receiver transmits an NR-SACK instead of a SACK. NR-SACKs are formatted as:

N : CumAckTSN; NRGapAckStart-NRGapAckEnd

The first NR-SACK (N:10;2–2) is also a dupack. It cum-acks TSN 10, and nr-gap-acks TSN 12. Once the sender learns that TSN 12 is non-renegable, the sender frees up kernel memory allocated to TSN 12, and the sending application writes more data into the newly available send buffer space. Since TSN 12 is also gap-acked (remember, every nr-gap-ack also gap-acks), the amount of outstanding data (O) is reduced to 7, allowing the sender to transmit new data, TSN 19.

Similarly (N:10;2–3) and (N:10;2–4) nr-gap-ack TSNs 12–13 and 12–14, respectively, and the sender removes TSNs 13 and 14 from the RtxQ. The sender transmits new data TSNs 20 and 21. On receipt of (N:10;2–4), the third dupack, the sender halves the cwnd ($C = 4$), fast retransmits TSN 11, and enters fast recovery. Dupacks received during fast recovery nr-gap-ack TSNs 15–20. The sender frees RtxQ

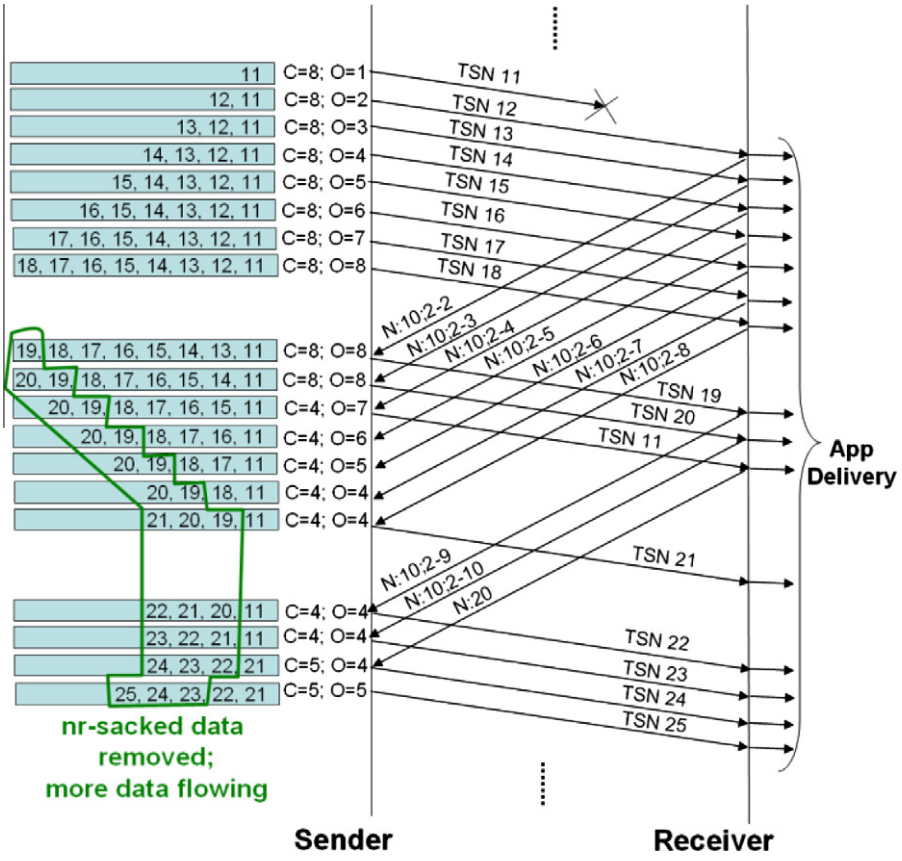

Fig. 4. Unordered SCTP data transfer using NR-SACKs.

accordingly, and transmits new TSNs 21, 22 and 23. The sender exits fast recovery when the NR-SACK with a new cum-ack (N:20) arrives. This new cum-ack increments $C = 5$, and decrements $O = 3$. The sender now transmits new data TSNs 24 and 25.

The explicit non-renegable information in NR-SACKs ensures that the RtxQ contains only *necessary* data - TSNs that are either in flight or "received and renegable". Comparing Figs. 2 and 4, we observe that NR-SACKs use the RtxQ more efficiently and achieve greater throughput.

3.3. Send buffer blocking

During an SCTP data transfer, the send buffer size imposes a hard limit on the size of the RtxQ. The data sender experiences send buffer blocking when the cwnd allows for sending out new data, but the send buffer consists of only the RtxQ, hence no new data can be transmitted until some data in the RtxQ is cum-acked. With NR-SACKs, the RtxQ is minimized allowing a sender to transmit more new data thus increasing the overall throughput.

We expect that, as long as the cwnd can grow up to the send buffer size, SCTP data transfer using SACKs will experience reduced throughput due to some level of send buffer blocking when there is loss. Although two different loss patterns with the same average loss can result in different cwnd evolution, we expect that in general for lower average loss rates, cwnd is more likely to grow as large as the send buffer size, which increases the chances of send buffer blocking with SACKs.

4. Simulation design

The open distribution of the ns-2 [5] SCTP and CMT modules were extended to transmit and process NR-SACKs at the data receiver and data sender, respectively. For SCTP tests (topology 1 in Fig. 5), a single path with various bandwidth-delay combinations has been configured. The performance of NR-SACKs has been tested for {8K, 32K, 64K, 128K, 256K} send buffer sizes in SCTP.

For CMT (topology 2 in Fig. 6), the sender and the receiver are multihomed, and CMT data transfer is carried out through two independent paths (Path 1 and Path 2). For all CMT experiments, both paths are simulated to have the same bandwidth (100 Mbps) and the same one-way delay {5, 45, 450} ms. On Path 1, each and every packet transmitted is independently lost with a probability

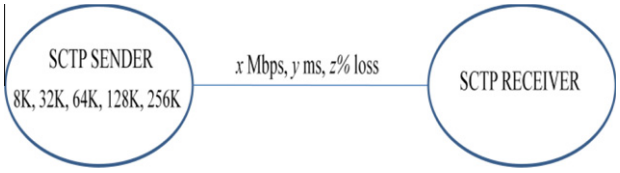

Fig. 5. Topology for SCTP experiments (topology 1).

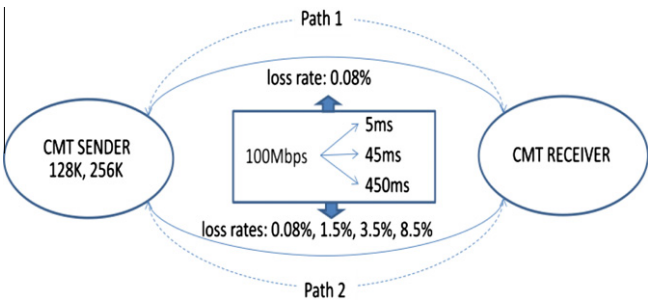

Fig. 6. Topology for CMT experiments (topology 2).

of 0.0008 (0.08%), whereas Path 2 is configured to have four different loss rates {0.08%, 1.5%, 3.5%, 8.5%} to investigate the effects of asymmetric paths. The performance of NR-SACKs has been tested for {128K, 256K} send buffer sizes in CMT.

The SCTP and CMT data transfers are *unordered* data transfers, as might be used by a disk to disk copy application. When all data is unordered, we expect the best possible gains using NR-SACKs since all out-of-order data is assumed delivered to the application immediately upon arrival, and therefore non-renegable.

5. Simulation results

Prior work [1] discussed throughput improvements achieved by using NR-SACKs in CMT, but did not present any SCTP throughput improvements, the reason being that the particular bandwidth-delay parameters studied showed limited throughput improvements for NR-SACKs over SACKs in SCTP. Our further investigation here shows scenarios do exist where NR-SACKs provide significant throughput improvement for SCTP. The scenarios previously investigated had low, mild, medium, and heavy cross traffic loads (<0.1%, 1–2%, 3–4%, 8–9% loss rates, respectively) that demonstrated some throughput improvement only for send buffers $\leqslant$32K. Investigation of SCTP data transfer for a broader range of loss rates has shown that NR-SACKs can provide better throughput improvements for send buffers $\leqslant$32K, and have the potential to provide throughput improvement for even larger send buffer sizes (Fig. 7).

To evaluate NR-SACKs vs. SACKs, we define *throughput gain* as $(T_{NR} - T_S) \times (100 \div T_S)$ where T_S is the throughput achieved by SACKs and T_{NR} is the throughput achieved by NR-SACKs for an identical set of experimental parameters (send buffer size, loss rate, bandwidth, and delay).

For presentation purposes, Fig. 7 uses a mixed scale. The throughput gain values for 32K and 64K send buffers are shown using the primary axes (bottom and left), and the values for an 8K send buffer are shown using secondary axes (top and right).

Mechanisms, such as congestion control and flow control, perform identically for NR-SACKs and SACKs. Therefore we hypothesized that NR-SACKs will always perform at least as well as SACKs. Fig. 7 confirms this hypothesis for all parameter combinations tested. In no case, on average, did SACKs outperform NR-SACKs.

More importantly, under certain conditions, NR-SACKs achieve a higher throughput than SACKs. Deeper investigation showed that this throughput improvement occurred as a result of send buffer blocking. We determined that, for send buffer blocking to occur, the loss rate must be below a certain level such that the congestion window, and hence RtxQ, can grow large enough to fill the entire send buffer. No send buffer blocking is expected if the loss rate is so high that cwnd is prevented from reaching the send buffer size.

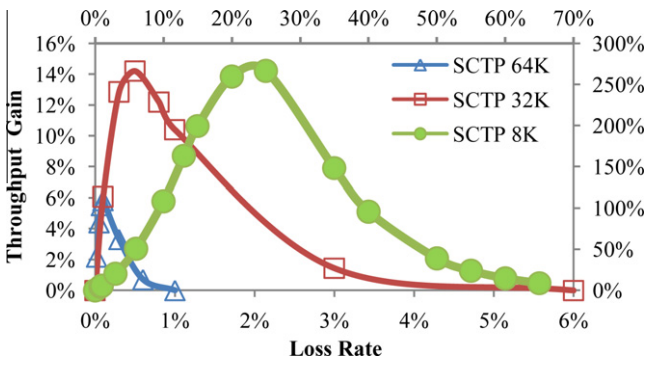

Fig. 7. SCTP throughput gain with NR-SACKs (45 ms, 100 Mbit).

In SCTP, without loss, no data will be received out-of-order; hence no gaps will be reported by both SACKs and NR-SACKs. Therefore, regardless of send buffer size, no improvement is expected with 0% loss as confirmed by Fig. 7. If only one data packet is lost after the data transfer is stabilized, exactly one instance of send buffer blocking will occur where NR-SACKs allow the sender to transmit some extra data due to the removal of non-renegable data. As more losses occur, NR-SACKs provide additional throughput improvement until the point where the loss rate becomes sufficiently high that cwnd cannot grow large enough to fill the entire send buffer, thus decreasing number of send buffer blocking instances.

5.1. Region of gain

For a given bandwidth-delay combination we define the *region of gain* as the loss rate interval, [*a*–*b*], where any loss rate between *a* and *b* results in an expected throughput gain of at least *x*, where *x* is the percentage improvement considered significant by the user. For our presented results, we chose *x* = 5%. For a given *x*, the interval [*a*–*b*] depends on the size of the send buffer.

Since higher loss rates result in smaller cwnds, we hypothesized that for smaller send buffers, more significant send buffer blocking would be observed at higher loss rates than for larger send buffers. Therefore, for a given bandwidth-delay combination, the right edge of the region of gain is expected to be a higher loss rate for smaller send buffer sizes.

This hypothesis is confirmed by Fig. 7. Even with loss rates as high as 65% (clearly an absurd loss rate in practice), NR-SACKs still provide significant throughput gain for an 8K send buffer which suggests that even a 65% loss rate allows the RtxQ to grow to 8K, thus blocking the send buffer. On the other hand, for 32K and 64K send buffers negligible throughput gain is expected when the loss rate is higher than 6% and 1%, respectively.

For a 100 Mbps link with 45 ms delay, the loss rate intervals [1–65+%], [0.09–2%], and [0.09–0.16%] are the approximate regions of gain for 8K, 32K, and 64K send buffers, respectively (Fig. 7). The peak throughput gain provided by NR-SACKs increases as the send buffer size decreases, the peak gain is >250% for 8K and ~6% for 64K (refer to the upper axis for 8K.) The region of gain narrows for larger send buffers, and widens for smaller send buffers.

Although send buffer blocking with SACKs appears to give NR-SACKs an opportunity to provide better throughput than SACKs, it would be incorrect to simply conclude that having increased number of send buffer blocking instances with SACKs will increase the gain from using NR-SACKs. For example, in Fig. 7, for an 8K send buffer, more send buffer blocking is expected at 20% loss than is expected for loss rates <20%. However the throughput gain NR-SACKs provide is higher at 20% loss than loss rates <20%.

5.2. Retransmission queue evolution

To gain insight into NR-SACKs, consider the Retransmission Queue (RtxQ) size over time. Fig. 8 shows the RtxQ size for an unconstrained send buffer during an SCTP data transfer using SACKs. The loss rate on the link is 10% which does not fall in the region of gain of a 32K send buffer with 45 ms delay and 100 Mbps bandwidth. Even though the send buffer is unconstrained, with 10% loss rate, RtxQ size never reaches 32K (Fig. 8). Therefore one can deduce that no send buffer blocking would occur for a 32K send buffer at 10% loss; hence NR-SACKs would provide no throughput gain under these conditions. For 45 ms delay and 100Mbps bandwidth, loss rates <6% do allow the cwnd to grow to 32K; hence NR-SACKs would provide throughput gain with loss rates <6% (Fig. 7).

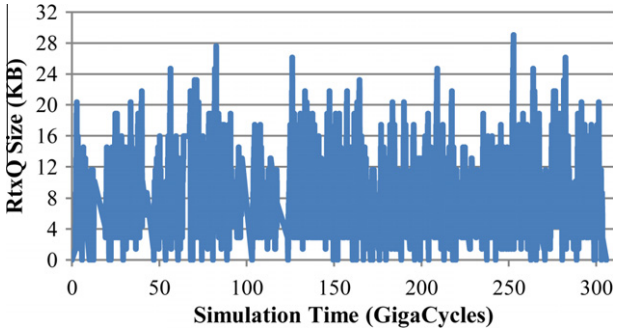

Fig. 8. Retransmission queue evolution for SCTP-SACKs (unconstrained send buffer, 10% loss, 45 ms, 100 Mbit).

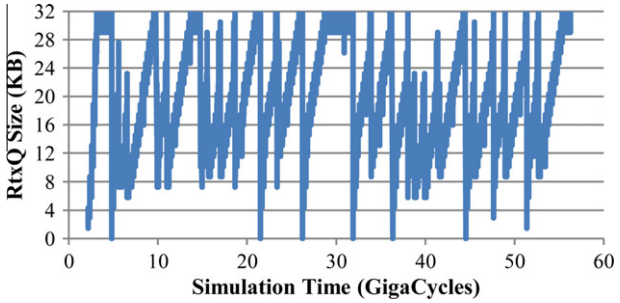

Fig. 9. Retransmission queue evolution for SCTP-SACKs (32K send buffer, 1% loss).

Fig. 9 shows a slice of one run for SACK 32K with 1% loss. This figure is characteristic of all of the runs under the same conditions. With 1% loss, as opposed to 10% loss, the RtxQ size does reach 32K several times. Each time the RtxQ = 32K is an instance of send buffer blocking. At the times where the RtxQ = 32K for SACKs, NR-SACKs can take advantage of the removal of non-renegable data from the RtxQ to allow the sender to transmit new data, thus achieving a better overall throughput.

NR-SACKs improve throughput by removing non-renegable data from the RtxQ during loss recovery, thereby allowing new data to be sent out during what would have been send buffer blocking with SACKs. Longer loss recovery periods help NR-SACKs provide higher throughput improvement. For a particular instance of send buffer blocking, if no data within the current send window is lost, then a data sender using NR-SACKs would not be able to remove any data from the RtxQ, and the data sender would not send out more data until a cum-ack was received (thus behaving no different than SACKs).

For NR-SACKs, if no data within the send window is lost (i.e., no gap in send window) for any send buffer blocking instances throughout a data transfer, no throughput improvement will be observed. So it is more accurate to say that "the throughput gain with NR-SACKs over SACKs is more significant as the ratio p/q increases where: p is the total loss recovery duration where send buffer blocking is occurring, and q is the total time for data transfer."

5.3. Impact of bandwidth and delay

Our prior work [1,2] did not evaluate the impact of different bandwidth-delay combinations on the performance of NR-SACKs vs. SACKs. Our investigation here shows that for a given delay the throughput gain provided by NR-SACKs does not vary with bandwidth when the Bandwidth-Delay Product (BDP) > send buffer size (upper four curves in Fig. 10). For links with a 45 ms delay and {3, 10, 100, 1000} Mbps bandwidths, the throughput gain graphs

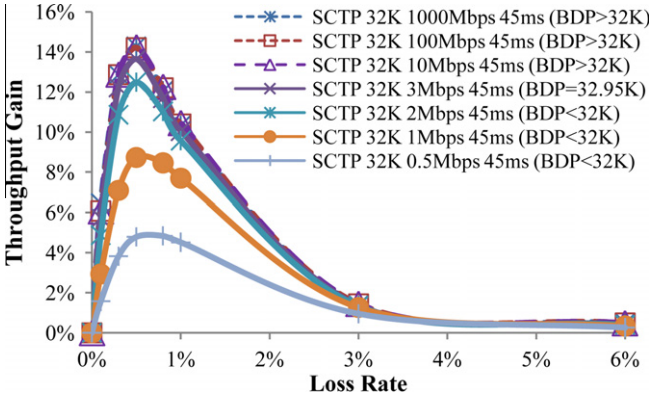

Fig. 10. NR-SACK throughput gain in SCTP for 32K send buffer, 45 ms delay, and various bandwidth parameters.

overlap for all the loss rates considered. In these cases, the BDP > send buffer size.

When the BDP < send buffer size, the send buffer size is no longer the limiting factor for the max cwnd as the cwnd cannot grow beyond the BDP. Since the cwnd is bound by the BDP, as BDP decreases the RtxQ becomes less likely to fill the entire send buffer. Therefore as BDP gets smaller, fewer send buffer blocking instances are expected, thus lowering the throughput gain provided by NR-SACKs (lower three curves in Fig. 10).

Fig. 11 shows how the throughput gain with NR-SACKs varies for a 32K send buffer under 100 Mbps bandwidth and various delay parameters. Note that for all the scenarios shown in Fig. 11, the BDP >32K. For all simulated loss rates, the gain is greater over a link with a shorter delay. We conclude that the region of gain for SCTP 32K gets wider as the delay gets shorter. The peak throughput gain significantly increases as the delay gets shorter. For a 100 Mbps link the peak gain for 32K send buffer is ~18% for 45 ms delay, and ~75% for 5 ms delay.

Although Figs. 10 and 11 show results for a 32K send buffer, we observed similar trends for different send buffer sizes (graphs not shown). That is, for a particular send buffer size, throughput gain; (i) does not change with bandwidth as long as the BDP > send buffer size, (ii) decreases as BDP gets smaller than send buffer size, and (iii) increases as the delay gets shorter regardless of BDP.

5.4. Impact of send buffer size

The default STCP send buffer sizes used in {FreeBSD 7.0, Solaris 10, Linux Ubuntu-2.6.31-i686} are {227K, 100K, 112K}, respectively.

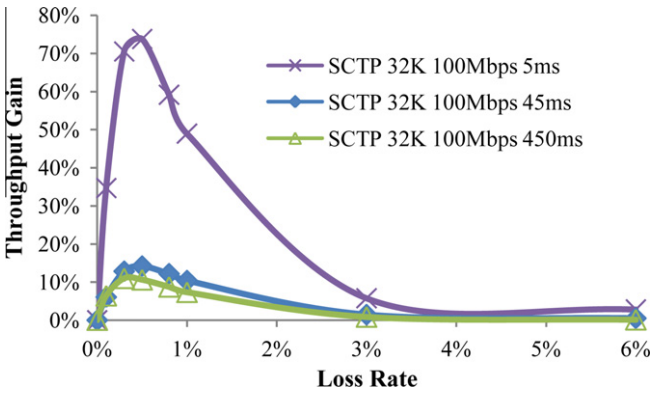

Fig. 11. NR-SACK throughput gain in SCTP for 32K send buffer, 100 Mbps and various delay parameters.

These SCTP implementations are still in development, so the default sizes remain open to change. Also the default send buffer size setting is configurable, so it is possible that systems with limited memory, such as handheld devices, will want to tune the send buffer size down to 32K or 64K.

Fig. 12 shows that NR-SACKs can provide throughput improvement for larger send buffers. With a 128K send buffer, a region of gain exists with the loss rate interval of [0.004–0.095%]. NR-SACKs provide ~14% throughput gain with 0.03% loss for 128K send buffer and a 5 ms delay. Even with a 256K send buffer, NR-SACKs achieve ~7% better throughput than SACKs for the loss rates [0.003–0.012%]. Although Fig. 12 shows the throughput gain for a 1 Gbps link, this bandwidth was chosen so that the BDP ⩾ send buffer size for all three send buffer sizes tested. The throughput gain is expected to be the same for lower bandwidths as long as the BDP ⩾ send buffer size as demonstrated in Fig. 10.

To provide intuition about the throughput gain that NR-SACKs might provide in a real life scenario, we ran an anecdotal experiment pinging http://www.youtube.com from a WiFi-enabled laptop connected to the University of Delaware's wireless network. Pinging the YouTube server 10,000 times, we observed an average round-trip delay of 8 ms and 101 lost packets, roughly ~1%.

We then ran an ns-2 experiment with these delay and loss values for wireless links of both 11 Mpbs and 54 Mbps. As an example, with a 32K send buffer NR-SACKs achieved throughput gains of ~37%, and ~60%, respectively. Note that the BDPs for the 11 Mbps and 54 Mbps links are ~10.7K, and ~52.7K, respectively. Since the BDP for the 11 Mbps link is less than the simulated send buffer size, the throughput improvement for 11 Mbps link is less than the observed gain of ~49% for 32K send buffer, 5 ms delay, and 100 Mbps shown in Fig. 11 of Section 5.2.

On the other hand, since BDP >32K for both 100 Mbps and 54 Mbps links, and 54 Mbps link has a slightly shorter delay (4 ms vs. 5 ms), the throughput improvement for the 54 Mbps is higher than that of 100 Mbps supporting the results shown in Fig. 11 (lower delay provides better gain as long as BDP ⩾ send buffer size).

In addition to potential throughput gain NR-SACKs provide, our results also suggest that NR-SACKs can achieve the same throughput as SACKs using smaller send buffers. Fig. 13 shows that NR-SACKs with a smaller send buffer, hence less memory can provide the same throughput as or a better throughput than SACKs with a larger send buffer. NR-SACKs with a 230K send buffer outperforms or achieves comparable throughput as SACKs with a 256K send buffer for the loss rates considered in Fig. 13.

By using NR-SACKs instead of SACKs, a highly loaded web server handling hundreds of thousands of simultaneous transport connections can potentially handle (i) the same number of connections using 10% less memory, or (ii) 10% more connections using the same amount of memory.

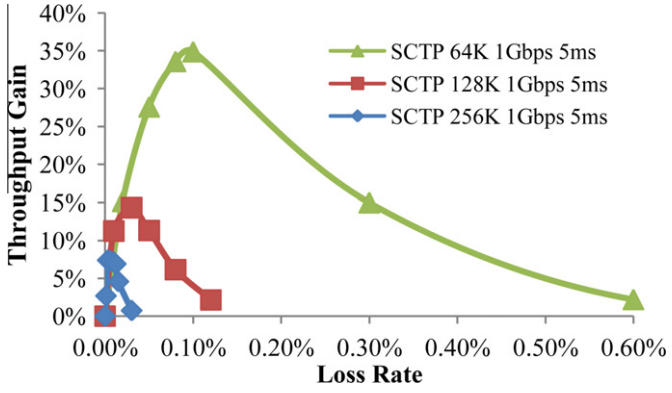

Fig. 12. NR-SACK throughput gain in SCTP (1 Gbps, 5 ms).

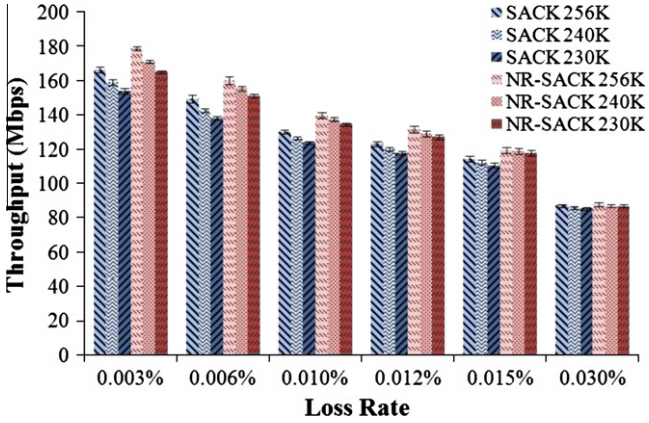

Fig. 13. SACK vs. NR-SACK throughput in SCTP (1 Gbps, 5 ms).

5.5. Impact of short delay and low loss rate

Some of the experimental parameters such as short delay and low loss rate used for some of our results might seem impractical. However, such short delays and low loss rates are observed in today's Internet. Consider increasingly popular content distribution networks (CDN). CDNs provide enhanced web browsing experience through their distributed servers strategically deployed around the world which are used to cache highly demanded web content. When web content is requested from a web server using CDN, using DNS redirection, the transport connection is opened to a physically closer CDN server; which increases the overall throughput by taking advantage of shorter delay and lower loss rate.

Based on an analysis conducted on Akamai [15], the largest CDN provider, in some cases transport connections experience delays in the order of a few milliseconds and loss rates as low as 0.001% [14]. Some of the well known servers using Akamai's CDN service are Oracle, Windows Update, Adobe, CBC, and MTV. Simply pinging http://www.oracle.com with 5,000,000 packets through an Ethernet interface connected to University of Delaware's network, we observed an average round-trip delay of 5.226 ms and a loss rate of 0.00166%. Depending on the send buffer size Akamai servers are using, NR-SACKs can provide throughput gain of 3–12% for the observed loss rate.

Unfortunately our ping experiments are anecdotal. YouTube and Akamai run over TCP not SCTP, and we were unable to determine their send buffer sizes. At this time, no commercial web servers are running over SCTP. We do observe that the default TCP send buffer sizes used in {FreeBSD 7.0, Solaris 10, Linux Ubuntu-2.6.31-i686, Mac OS X 10.2.0} are {32K,48K,16K,64K}, respectively. Our ping experiments suggest that when certain send buffer sizes are used, NR-SACKs could improve throughput in these and similar real life scenarios.

We note that if dynamic autotuning buffering management [12] is enabled such that none of the connections have send buffer limitation (unlikely for a highly loaded web server), then the benefits of NR-SACKs would only be for improved memory utilization, not throughput. It is important to emphasize that NR-SACKs will not decrease throughput.

5.6. Results for CMT throughput

Our prior work [1] did not investigate the effects of bandwidth and delay on potential throughput gains that NR-SACKs can provide to CMT. As was the case for SCTP, in CMT, when the BDP > send buffer size the bandwidth has no effect on the throughput gain. Path-1 in Fig. 6 is designed to have a simulated loss rate of 0.08% in all CMT experiments. The bandwidth and delay

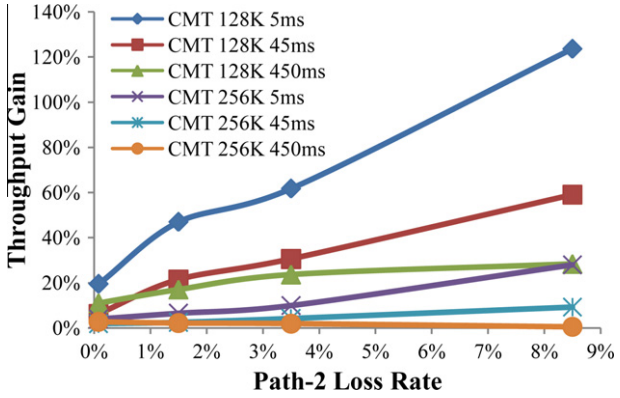

Fig. 14. NR-SACK throughput gain in CMT (100 Mbps, path 1 loss rate: 0.08%).

parameters for both paths are always identical. Fig. 14 shows the observed throughput gain through paths with {5,45,450} ms delays for 128K and 256K send buffers in CMT where the loss rate for the Path-2 ranges from 0.08 to 8.5%.

Regardless of the delay, for both 128K and 256K send buffers, the throughput gain increases as the loss on Path 2 increases. As the two paths become more asymmetric in terms of loss, more out-of-order data is received by the data receiver. The additional out-of-order data in turn increases the loss recovery durations where send buffer blocking is occurring. For all loss rate-delay combinations studied, the throughput gain is greater for smaller send buffers.

Delay has a strong impact on the performance of NR-SACKs in CMT (Fig. 14). The throughput gain achieved by NR-SACKs dramatically increases as the delay gets shorter. For a 128K send buffer, NR-SACKs provide a throughput gain of up to 120% (5 ms). In other words, when Path 2 experiences 8.5% loss, transferring a file through paths with 5 ms delay takes 2.2 times longer using SACKs than using NR-SACKs. For 5 ms delay, even when both paths have a loss rate of only 0.08%, the data transfer using SACKs takes 20% longer than the data transfer using NR-SACKs (leftmost data point in Fig. 14).

6. Recommended NR-SACK management

Since NR-SACKs are optional, the endpoints must first negotiate NR-SACK usage during association establishment as depicted in Fig. 15. An endpoint supporting the proposed extension lists NR-SACK in the Supported Extensions Parameter carried in the INIT or INIT ACK RFC5061 [13]. During association establishment, if both endpoints support the NR-SACK extension, then each endpoint acknowledges received data with NR-SACKs, not SACKs.

Once the use of NR-SACKs is negotiated, both endpoints take the following actions for correct operation of NR-SACKs throughout the data transfer.

- when out-of-order data is received, the data is classified as either *renegable* or *non-renegable*. An out-of-order data is classified as non-renegable by the data receiver (1) if the receiver never reneges (as in FreeBSD when an endpoint sets *sctp_do_drain = 0*), or (2) immediately after the received data is delivered to the application (for example, when the out-of-order data is unordered or in-order within the stream).
- when an NR-SACK is to be sent, *renegable* out-of-order data are reported in r-gap-ack block(s), and *non-renegable* out-of-order data are reported in nr-gap-ack block(s).
- when an NR-SACK is received at the data sender, the r-gap-ack and nr-gap-ack blocks are processed as would be gap-ack blocks for a regular SACK. In addition, all nr-gap-acked (non-renegable) data is removed from the data sender's retransmission queue.

7. Conclusions

Our SCTP and CMT throughput analysis shows that the throughput with NR-SACKs is never worse than SACKs, and for certain regions of gain significantly better. The trade-off of using NR-SACKs is the additional processing time required to fill and process the new fields in an NR-SACK at the data receiver and data sender, respectively, and a few more bytes on the wire. Based on our ns-2 NR-SACK module, we believe these overheads to be negligible.

To make NR-SACKs available in practice, we are currently implementing them in FreeBSD 7.0. Once implemented, NR-SACKs will be evaluated in a variety of network environments using an emulation test-bed based on Dummynet [3] as shown in Fig. 16.

The concept of NR-SACKs can be applied to any reliable transport connection that either: (1) permits out-of-order data delivery, or (2) by OS design, does not permit a data receiver to renege. In regards to (2), while reneging is permitted, it should only occur during drastic situations, such as when an OS needs to reclaim previously allocated buffer space to keep a system from crashing. We are currently investigating how frequently, if at all, reneging occurs in today's Internet by analyzing TCP traffic traversing several major

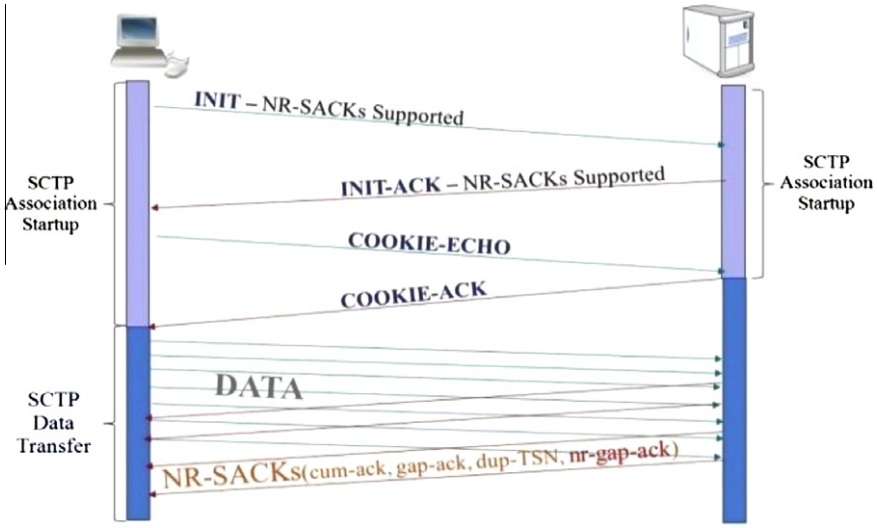

Fig. 15. NR-SACK negotiation in SCTP.

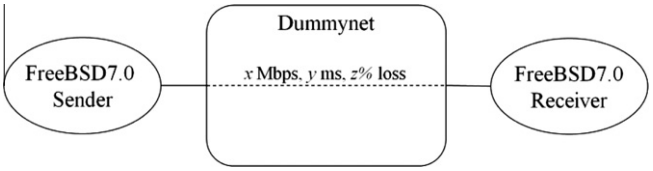

Fig. 16. Emulation experimental design.

routers, and by checking if today's operating systems are programmed to renege and/or handle reneging [8]. Our reneging investigation will provide insights and empirical evidence to argue for the total replacement of SACKs with NR-SACKs.

References

[1] P. Natarajan, N. Ekiz, E. Yilmaz, P. Amer, J. Iyengar, R. Stewart, Non-Renegable Selective Acknowledgments (NR-SACKs) for SCTP, in: International Conf on Network Protocols (ICNP), Orlando, October 2008.
[2] P. Natarajan, Leveraging Innovative Transport Layer Services for Improved Application Performance, PhD Dissertation, Computer & Information Sciences Department, University of Delaware, USA, 2009.
[3] L. Rizzo, Dummynet: a simple approach to the evaluation of network protocols, ACM Computer Communication & Review 27 (1) (1997) 31–41.
[4] P. Natarajan, P. Amer, E. Yilmaz, R. Stewart, J. Iyengar, Non-Renegable Selective Acknowledgments (NR-SACKs) for SCTP, IETF Internet Draft, work in progress, January 2010.
[5] N. Ekiz, P. Natarajan, J. Iyengar, A. Caro, ns-2 SCTP Module, Version 3.7, September 2007. <http://pel.cis.udel.edu/>.
[6] J. Iyengar, P. Amer, R. Stewart, Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths, IEEE/ACM Transaction on Networking 14 (5) (2006).
[7] FreeBSD TCP and SCTP Implementation. <http://www.freebsd.org/cgi/cvsweb.cgi/src/sys/netinet/#dirlist>.
[8] N. Ekiz, Transport layer reneging, PhD Dissertation, CISC Department, University of Delaware (in preparation).
[9] D. Wischik, M. Handley, M. Braun, The Resource Pooling Principle, Computer Communications Review 38 (5) (2008) 47–52.
[10] M. Mathis, J. Mahdavi, S. Floyd, A. Romanov, TCP Selective Acknowledgment Options RFC2018 (1996). <http://www.rfc-editor.org/rfc/rfc2018.txt>.
[11] R. Stewart et al., Stream Control Transmission Protocol RFC4960 (2007). <http://www.rfc-editor.org/rfc/rfc4960.txt>.
[12] J. Semke, J. Mahdavi, M. Mathis, Automatic TCP Buffer Tuning, in: Proc. ACM SIGCOMM, Vancouver, September 1998.
[13] R. Stewart et al., Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration, RFC5061, September 2007. <http://www.rfc-editor.org/rfc/rfc5061.txt>.
[14] A. Su, D. Choffnes, A. Kuzmanovic, F. Bustamante, Drafting behind Akamai: Inferring network conditions based on CDN redirections, IEEE/ACM Trans. Netw. 17 (6) (2009) 1752–1765.
[15] Akamai Technologies, Inc. <http://www.akamai.com/>.