BBN Technical Report No. 8333

Explicit Transport Error Notification (ETEN) for Error-Prone Wireless and Satellite Networks

Job No. 12318220

February 8, 2002

(Revised: March 22, 2002)

Prepared for:

NASA Glenn Research Center
21000 Brookpark Road, MS 54-5
Cleveland, OH 44135

Prepared by:

BBN Technologies
10 Moulton Street
Cambridge, MA 02138

.

Explicit Transport Error Notification (ETEN) for Error-Prone Wireless and Satellite Networks*

Rajesh Krishnan, Mark Allman, Craig Partridge, and James P.G. Sterbenz †
{*krash,mallman,craig,jpgs*}*@bbn.com*

March 22, 2002

Abstract

Wireless and satellite networks have non-negligible error rates that can significantly influence TCP performance because TCP considers every packet loss as an indicator of congestion, and thus throttles the packet transmission rate. Explicit transport error notification (ETEN) mechanisms can aid TCP in distinguishing packets that are lost due to congestion from ones that are lost due to corruption. If TCP can retransmit a packet lost due to corruption without needlessly reducing the transmission rate, a performance benefit can be realized.

In this study we propose two types of ETEN mechanisms: (i) *per-packet* mechanisms that notify endpoints of each detected corruption; and (ii) *cumulative* mechanisms that notify endpoints of aggregate corruption statistics. We have implemented the proposed mechanisms in the ns-2 simulator. We present simulation results on performance gains achievable for TCP Reno and TCP SACK, using ETEN mechanisms over a wide range of bit error rates and traffic conditions. We compare TCP Reno and TCP SACK enhanced with ETEN mechanisms against TCP Westwood, which uses a bandwidth estimation strategy in place of the traditional AIMD congestion avoidance algorithm. We discuss two issues related to the practical deployment of ETEN mechanisms: corruption detection mechanisms (and their co-operation with ETEN-based recovery in the transport layer) and security aspects. We include recommendations for further work. Our conclusions from this study are:

1. per-packet ETEN mechanisms offer substantial gains in bulk TCP goodput in the absence of congestion; however, in the presence of congestion TCP congestion avoidance mechanisms dominate resulting in insignificant gains from ETEN
2. the proposed per-packet mechanisms provide useful upper bounds on performance that can be used to evaluate future proposals of per-packet and cumulative ETEN techniques
3. per-packet mechanisms present significant challenges to practical implementation by providing a new opportunity to exploit Internet security vulnerabilities and by requiring intermediate nodes to reliably extract information from the headers of corrupted packets
4. cumulative ETEN techniques are more attractive to implementation; however, the particular mechanism we evaluated did not realize the potential gains of per-packet techniques
5. future work in this area should focus on alternative cumulative ETEN mechanisms, accurate loss inference at endpoints to avoid tracking congestion losses at every hop, interactions with forward error correction, and cross-layer co-operation for ETEN

Keywords: explicit transport error notification, wireless and satellite networks, TCP performance, congestion, corruption, bit errors.

*This work was done by BBN Technologies under contract number NAS3-99175 from NASA Glenn Research Center and NASA's Earth Science Technology Office.

†The authors wish to thank Will Ivancic of NASA Glenn Research Center for his valuable input throughout this effort and for his insightful reviews that have significantly improved this document.

.

Contents

1 Introduction

One obstacle to good performance of internetworks with wireless and satellite components is non-negligible bit-error rates (BER). The most widely used transport protocol in the TCP/IP suite, the Transmission Control Protocol (TCP) [27], guarantees that corrupted data will be retransmitted by the data sender, hence providing a reliable byte-stream to applications. However, packet loss is also used by TCP to determine the level of congestion in the network [10] – as traditionally, the bulk of packet loss in networks comes from router queue overflow (i.e. congestion). Therefore, to avoid congestion collapse TCP responds to packet loss by decreasing the congestion window [10] [33], and therefore the sending rate. The reduction of the congestion window is not needed to protect network stability in the case when losses are caused by corruption and therefore these needless reductions in the sending rate have a negative impact on performance with little overall benefit to the network.

If the TCP sender can distinguish packets lost due to congestion from packets lost due to errors[1], better performance may be achieved. The performance benefit can be realized if TCP can retransmit a packet lost due to corruption without needlessly reducing the transmission rate, while continuing to protect network stability by decreasing the sending rate when loss is caused by network congestion.

Several approaches have been proposed in the literature to distinguish congestion losses from errors. For example, if explicit congestion notification (ECN) [6] [15] [32] [37] is used, it may be possible to implicitly determine some (but not all) losses due to corrupted packets using some as-yet-to-be-developed heuristics. Previous attempts to implicitly distinguish error from congestion have not been successful [3] [36]. While Performance Enhancing Proxies (PEPs) [35] can be used to improve TCP performance, they break the end-to-end semantics of the transport layer connection. Therefore, it is desirable to evaluate explicit transport error notification (ETEN) mechanisms. For earlier work on explicit loss notification in the context of TCP over wireless and satellite links, see [1] [2] [17] [18]. An analysis of situations that can benefit from ETEN mechanisms is given in [36].

The purpose of this study is two-fold:

1. To establish bounds on the performance improvements that can be obtained with the use of ideal ETEN mechanisms under different network conditions – error rates, capacities, delays, topologies, congestion – and thereby determine promising directions for future research, if any.

2. To consider issues related to practical deployment of ETEN mechanisms, to propose suitable architectures and mechanisms, to identify security vulnerabilities, and to identify areas that require further study before an ETEN system is viable.

Through simulations, we have evaluated possible enhancements to TCP that are based on ETEN notifications from intermediate routers and/or end systems. Emulations in a testbed and live testing over real networks were considered out of scope of this effort. This study included the following tasks:

- Determine bounds on TCP goodput[2] improvements possible from ETEN when a TCP sender is presented with ideal information about the cause of each loss.

[1] The term "errors" in this document is used to refer specifically to link and router errors [23] that cause corruption of the packet headers and/or data.

[2] The *goodput* of a flow is defined as the bandwidth delivered to the receiver, excluding duplicate packets [7]. We calculate the goodput by dividing the total number of unique bytes arriving at the receiver by the duration of the TCP connection (Note: the header bytes of these unique packets are also included).

- Evaluate via simulations, actual performance achievable over a range of network topologies and traffic conditions with different TCP variants such as Reno and SACK.
- Discuss and evaluate the performance of specific ETEN mechanisms that fall in one or more of the following broad categories:
  - Forward notification – whereby any notification about corrupted packets is sent in the direction of the data packets and then returned to the sender in TCP acknowledgment segments.
  - Backward notification – in which a message is sent from the node (end-host or intermediate router) that detects a corrupted packet to the host that originated the packet.
  - Per-packet mechanisms that attempt to determine the root cause of each loss experienced.
  - Aggregate notification schemes where the TCP sender is provided with aggregate statistics about the loss patterns experienced in the network path.
- Determine how TCP should best react upon receiving ETEN notification.
- Assess the security implications of introducing various ETEN mechanisms into the Internet architecture. These include:
  - Potential vulnerabilities of the proposed mechanisms to distributed denial-of-service attacks.
  - Operation over encrypted tunnels, VPNs, and MPLS paths, where intermediate nodes may not be able to determine actual source or destination IP addresses and ports, making ETEN notification effectively impossible.
  - Vulnerabilities to mis-behaving receivers that attempt to mask congestion-related losses using ETEN mechanisms in an attempt to obtain an unfair share of network resources.

The rest of this document is organized as follows. In **Section 2** we describe different explicit error notification mechanisms and appropriate sender responses to the notifications. In **Section 3**, we describe performance results from the simulation of two specific per packet ETEN mechanisms, namely, the Oracle ETEN and BETEN described in Section 2. Results include cases of single and multi-hop topologies, with and without cross traffic. **Section 4** describes the implementation and performance evaluation of a cumulative ETEN mechanism, in which aggregate error statistics for the path are used to distinguish between corruption and congestion on a probabilistic basis. **Section 5** describes the potential to dynamically adjust Forward Error Correction (FEC) code strength based on cumulative path error statistics. In **Section 6** we outline different mechanisms that can be used by routers and end-systems to detect corrupted packets. In **Section 7** we discuss potential security vulnerabilities for ETEN mechanisms. A concluding summary is presented in **Section 8** with recommendations for future work. Bibliographic references are included at the end. Additional plots from the simulations are included in the appendices for completeness.

2 Error Notification and Response Mechanisms

In this section, we present protocol mechanisms to explicitly notify hosts of errors, and we discuss response strategies that can be adopted by the sender. For the mechanisms proposed in this report we assume one of the following two cases holds:

1. The source and destination IP addresses, the source and destination TCP ports, and the TCP sequence number can be correctly obtained from the corrupted packet. In addition, the packet in question must

be part of the sender's current window; otherwise, the opportunity to mitigate the performance problems caused by the corrupted packet is lost.

2. The node detecting errors can only calculate cumulative error rates for each link. In other words, the information in the header of a corrupted packet is considered inaccurate.

A middle ground between the above two scenarios is possible. That is, *some* of the information in the header is accurate. In this case, information about specific corrupted packets can be propagated to the host that is believed to have sent the packet. That host can then apply a set of heuristics in an attempt to match the corrupted packet to a specific TCP connection. For instance, consider the case when the IP addresses of a packet are not corrupted but one of the TCP port numbers is in error. An ETEN message arriving at source can be passed through a heuristic that attempts to match the corrupted packet to a known connection based on the accurate information in the packet. Designing effective heuristics will depend upon the specific error patterns that are observed, and leave this to future work.

When the information in the header of the corrupted packet is inaccurate, it is tempting to send notifications to both the sender and the receiver with the hope that at least one of them can match the correct flow and initiate recovery. However, if two notification messages are generated for each corrupted packet, this will increase the load on the network. If the error rates are high at a number of points in the network, the load from the additional notifications may result in congestion. Furthermore, the sender has to maintain additional state in this case to avoid sending two retransmissions for one corrupted packet.

Next, we consider the issue of propagating ETEN information to the TCP sender. We expect that the sender can use this information to make better congestion control decisions. We could create a new protocol to exchange ETEN messages, but it may be more desirable to extend existing protocols for this purpose. There are several possibilities. We can report ETEN information about specific corrupted packets using the Internet Control Message Protocol (ICMP) [26]. Cumulative error rates can also be reported using ICMP. We can use TCP header bits (for example, the reserved bits may be reallocated for this purpose). We can add ETEN information within an IP or TCP option. These methods can be used to mark subsequent packets in a flow to indicate previous losses due to packet corruption in the same flow. Alternatively, we can mark individual packets and pass them along rather than discard them when corruption is detected. However, this requires inter-layer co-operation, heuristics to determine the correct source and destination (since the header may be corrupted), and changes in IP router requirements [28] (to allow for packet repair).

Note that not all of the methods outlined above will work in all cases. For instance, adding information to a second packet in the same flow may not be possible when using IP layer security because it is difficult to determine which flow a given packet belongs to. A larger discussion of security issues with ETEN is given in Section 7.

In the study presented in this report we evaluated various methods to propagate information back to the sender. We have described each of the methods used as we discuss each class of ETEN mechanisms in the following subsections. The specifics of our implementation of these mechanisms is given in Section 3.

Finally, note that many link-layer protocols include a corruption detection mechanism (such as a CRC or a checksum). Therefore, corrupted packets may not get passed up to the higher layers, but rather silently get discarded by the link-layer. A number of the mechanisms described in the following subsections assume that there is a higher degree of interaction (for reporting errors) between the link-layer and the network layer than currently exists. While this is a significant assumption, the goal of the work presented in this report is to investigate the efficacy of ETEN without being constrained by current technology and protocol machinery[3]. Therefore, we leave this inter-layer communication as future work.

[3]We expect that most of our assumptions are implementable and not purely academic.

2.1 Sufficient information available about endpoints

We consider three different kinds of mechanisms when ideal knowledge of the IP addresses, TCP ports, and the TCP sequence number corresponding to the corrupted packet are available. The first of these, called *Oracle ETEN*[4], is an idealized case that allows instantaneous and perfect notification of errors. The other two schemes, *Backward ETEN* and *Forward ETEN*, use ICMP-like signaling to notify hosts of a corruption, and are analogous to forward and backward explicit congestion notification schemes.

2.1.1 Oracle ETEN

Oracle ETEN, illustrated in Figure 1, is a theoretical construct that assumes sufficient knowledge about the the corrupted packet (sender and destination IP addresses, sender and destination TCP port numbers, and the TCP sequence number) is available to the intermediate router or the end-system that detects corruption. Furthermore, this mechanism assumes that the source of the flow can be instantaneously notified of the packet corruption. Oracle ETEN provides an upper bound on the performance improvement achievable by ETEN mechanisms that notify the source. While the Oracle ETEN mechanism is an impossibility in the real-world, it can be used to distinguish between cases in which *some* ETEN mechanism would be useful and cases when no ETEN scheme would aid performance. We implemented Oracle ETEN in ns-2 TCP code; simulation results are presented in Section 3.

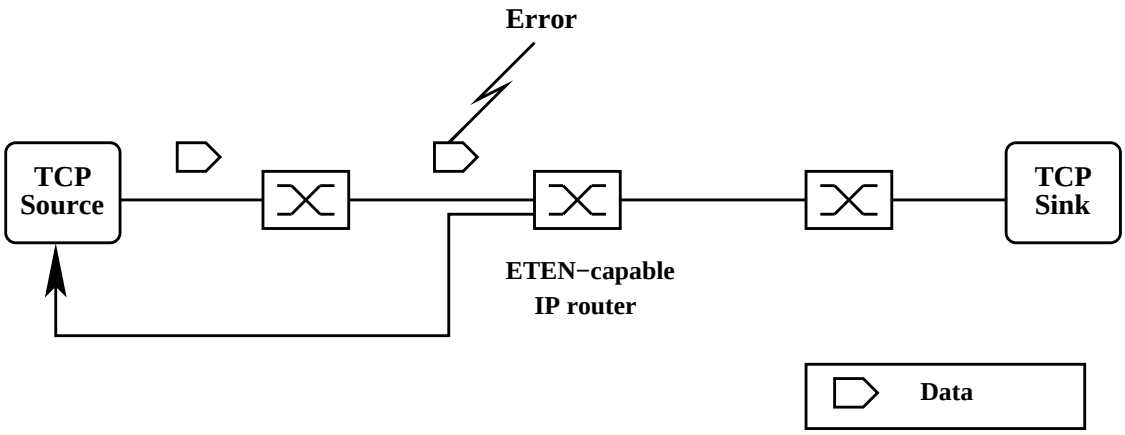


Figure 1: Oracle ETEN

2.1.2 Backward ETEN

The backward ETEN (BETEN) mechanism, illustrated in Figure 2, is analogous to backward explicit congestion notification schemes (e.g., source-quench [26]). This mechanism assumes that the intermediate router can extract or reconstruct (e.g., using FEC) sufficient knowledge about the corrupted packet that is required to notify the sender. We have implemented in ns-2, a BETEN mechanism in which an intermediate node transmits an explicit (ICMP-like) message to the sender upon detecting a corrupted packet. The BETEN mechanism we have implemented provides the best-case scenario in terms of the actual time required to notify the sender that a corrupted packet has been detected.

[4]Oracle ETEN is not an implementable mechanism, but rather it is a theoretical construct that can be simulated.

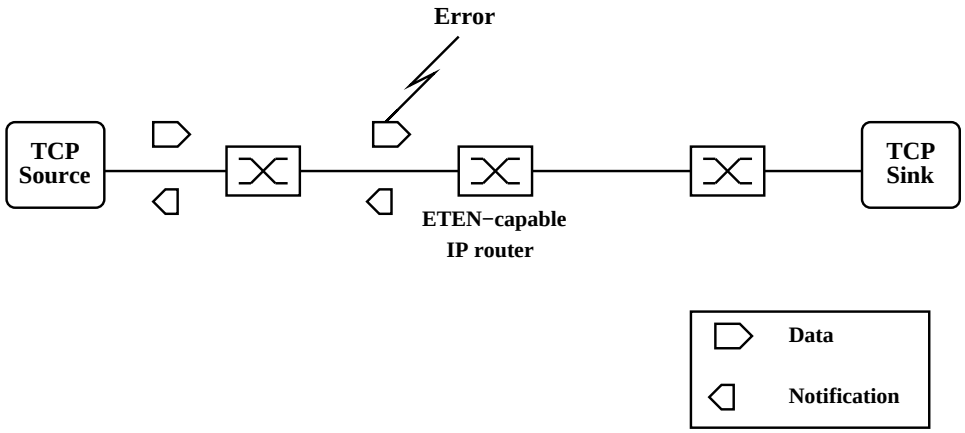


Figure 2: Backward ETEN

2.1.3 Forward ETEN

The forward ETEN mechanism illustrated in Figure 3 is analogous to forward explicit congestion notification schemes (e.g., [32] [37]). This mechanism also assumes that the intermediate router can extract (or reconstruct using FEC) complete and correct knowledge of the IP addresses, TCP ports, and TCP sequence number corresponding to the corrupted packet. Upon detection of a corrupted packet, the intermediate router transmits a FETEN message to the destination host, which then conveys the information to the sender on a subsequent acknowledgment.

If separate messages are generated per corruption loss, it is easy to see that BETEN will lead to faster recovery than FETEN. The benefit of using BETEN is higher if the corruption occurs closer to the sender and it increases with the round-trip delay of the path.

Two alternatives that do not require generation of new packets for FETEN exist. With the first alternative, the intermediate node that detects the corrupted packet will wait for a subsequent (uncorrupted) packet from the corresponding flow and piggyback the corruption information on that packet. The other approach is to mark the corrupted packet and pass it along to the destination (subsequent nodes must also forward this packet). The destination in turn will notify the sender of the packet lost due to corruption. As noted earlier, this last option requires requires inter-layer co-operation, heuristics to determine the correct source and destination (since the header may be corrupted), and changes in IP router requirements (to allow for packet repair).

2.2 Insufficient information available about endpoints – Cumulative ETEN

In practice, we cannot always accurately retrieve the source and destination IP address, source and destination TCP port numbers, and TCP sequence number from a corrupted packet or link-layer frame. Therefore, in this section we consider ETEN mechanisms that work on the basis of cumulative error rates (for example, error rates that are averaged over an interval of time and across various flows), rather than attempting to make notifications on a per-packet basis.

The cumulative ETEN (CETEN) information conveyed to the end-hosts can be in one of several different forms:

- An *absolute* bit error rate, byte error rate, or packet error rate observed within a moving window in time. The error rate may be quantized into a small number of steps (for example, *high*, *medium*, and *low*).

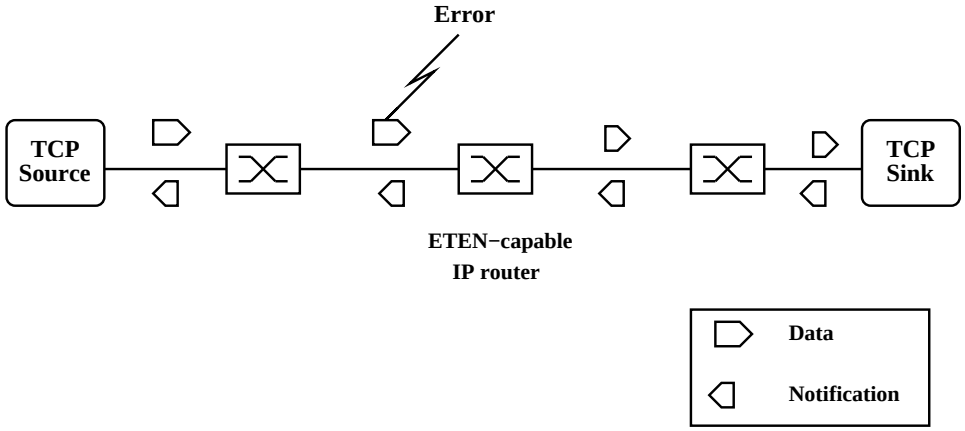


Figure 3: Forward ETEN

A binary feedback scheme [15] (see also [32] [37]) is a special case that provides indication that the bit/byte/packet error rate exceeds some threshold.

- A *relative* error rate that simply indicates that the quantized error rate has increased or decreased from the previous value.
- An estimate of the probability that a packet survives corruption.

CETEN information can be delivered to a sender via forward or backward signaling, analogous to a FETEN-based or a BETEN-based strategy. Also, as noted before, CETEN can be piggybacked on data and acknowledgment packets, rather than using additional distinct messages.

CETEN information can be collected on a per-hop basis or aggregated over the end-to-end path. Due to the difficulty in correctly assigning corrupted packets to their corresponding flows, any per-flow CETEN information has to be estimated, for example from what is observed across all flows using a given link. CETEN strategies that rely purely on statistics collected within the lifetime of a particular flow are of limited use for short flows. For example, a short flow may have terminated before we obtain a good estimate of the packet corruption probability.

We note that CETEN information can be used to control the strength of either per-packet FEC schemes or erasure codes. With CETEN, FEC strength adjustment can be done selectively on a per-flow basis. We will discuss the use of FEC in conjunction with CETEN in Section 5.

In Section 4, we will describe specific CETEN strategies that were explored (via simulations) within the scope of this study. Real-world implementation and analysis of CETEN is a subject for future work.

2.3 Sender response to ETEN

The sender's response to an ETEN notification depends on the type of the notification. If the sender receives timely and reliable information about the corrupted packet that identifies the TCP flow and the sequence number within the flow, then the sender can retransmit the corrupted packet without adjusting the congestion state.

However, if the information contained in the ETEN notification is only partially reliable, or if only a cumulative error rate is available, then the sender has to apply a heuristic to determine what action is appropriate. When a

transport endpoint infers a packet loss, it cannot exactly determine from the CETEN information if the packet loss occurred due to corruption or congestion. At best the CETEN information provides a recent estimate of the fraction of the losses that are due to corruption. The decision to be made by the sender includes whether an outstanding segment should be retransmitted and whether the congestion state should be altered in response.

Since most link level technologies require corrupted packets to be discarded even before it reaches the IP layer, per-packet ETEN mechanisms (at the IP and TCP layers) cannot see the corrupted packets. Although the sender response to per-packet ETEN is more straight-forward than the response to CETEN, it must be noted that the corruption link layer counters of errors are readily available; these counters can be used to generate CETEN.

3 Performance of Per-Packet ETEN Mechanisms

In this section, we describe results of simulations on the performance of Oracle ETEN and BETEN, both of which notify the TCP sender of each corrupted packet. Various types of links (e.g., terrestrial LAN, WAN, and satellite), modeled by their respective latencies, are simulated over a wide range of bit error rates. ETEN performance is compared against conventional Reno [10] and SACK [29] variants of TCP. We vary the following parameters (actual values used in the study are listed in Table 1):

- ETEN schemes – None, Oracle ETEN, and BETEN
- Link characteristics – bandwidth, delay, and error rate
- TCP congestion avoidance strategy – Reno, SACK

Oracle ETEN represents the ideal, yet impossible, baseline that provides an upper bound on the performance achievable by any practical per-packet ETEN scheme. One design goal is that the addition of any ETEN scheme (to any given TCP congestion avoidance strategy) should not make the performance worse; therefore, the case with no ETEN is expected to provide a useful lower bound (and, this is shown in our simulation results). The BETEN strategy represents an implementable per-packet ETEN strategy (assuming that we can extract sufficient information from corrupted packets). In the absence of congestion, we can expect that the goodput when using BETEN will lie between the goodputs using Oracle ETEN and no ETEN.

In this section, we consider six sets of simulations, as follows:

1. This set of simulations is aimed at evaluating the gains possible over a single uncongested link using Oracle ETEN and BETEN with TCP Reno and TCP SACK.
2. In this set of simulations we use a 3-hop linear topology of identical links, while varying the other parameters outlined above. These simulations serve the purpose of validating our implementation in a more complex topology with multiple links and routers. The results are expected to match those of the first set.
3. In this set of simulations, we use a 3-hop linear topology to provide insight into the performance of ETEN mechanisms in the face of congestion from constant-bit-rate UDP traffic. The intensity of cross-traffic is varied across simulation runs. The competing traffic in our simulations does not use a congestion avoidance strategy.
4. This set of simulations offers competing TCP traffic (instead of UDP traffic) and is otherwise identical to the third set. This provides insight into the performance of ETEN when the competing traffic flows use a congestion avoidance strategy also.

5. This set of simulations compares the performance of our ETEN mechanisms with TCP Westwood [12] in the absence of cross traffic.

6. This set of simulations compares the performance of our ETEN mechanisms with TCP Westwood in the presence of UDP cross traffic.

Parameter	Value
Link capacity	1.5 Mb/s, 10 Mb/s, 100 Mb/s
Forward link bit error rate	1.56×10^{-10} – 1.56×10^{-5}
Backward link bit error rate	0 or same as forward bit error rate
Link propagation delay	10ms, 100ms, 320ms
TCP variants	Reno, SACK
ETEN mechanisms	none, Oracle ETEN, BETEN
MSS	536 bytes
Receiver window	20 segments
Router buffers	50 packets shared FIFO queue

Table 1: Experiment Parameter Ranges

3.1 Statistical limitations of this study

Our goal in this study is to observe ETEN behavior over a broad sweep of the parameter space in order to determine general trends when using ETEN. Therefore, we limit each data point to a single TCP run – which may have led to some anomalies in the graphs (for example, drops in goodput can occur due to a large number of losses during slow-start in a particular random simulation). The plots in the appendices provide the reader with a general idea of how ETEN performs across the parameter space.

The simulation code is available to the research community for further experimentation [4]. This code enables researchers to explore other scenarios (for example, focus on a particular range of error rates).

We note the following statistical limitations of the study:

1. the TCP flows may not have a duration long enough for the flow to experience a sufficient number of corruptions; thus the actual value of goodput gains achieved using ETEN may not be statistically significant in some cases; however, in order to make this less likely, we have not included any data point in which the expected number of corruptions during the simulation duration is less than 25 if transmissions occurred at the line-rate (see Table 2 for error rates expected at various line-rates and BERs over 120 seconds)

2. each data point is based on the results of a single run; this is an issue especially since the state of TCP when errors occur (for example, three-way handshake, slow-start, or AIMD congestion-avoidance) will influence performance

3.2 Simulation software description

We extended the ns-2 simulator [14] (version 2.1b7a) to support our ETEN simulations. The following discussion assumes that the reader is familiar with ns-2.

Bit Error Rate Link Capacity (Mb/s)	1.56e-10	1.56e-9	1.56e-8	1.56e-7	1.56e-6	1.56e-5
100	1.872	18.72	187.2	1872	18720	187200
10	.1872	1.872	18.72	187.2	1872	18720
1.5	.02808	.2808	2.808	28.08	280.8	2808
.128	.0001997	.0019968	0.01968	0.1968	1.968	19.68

Table 2: Errors expected in 120s at various line-rates and various BERs

We have added a `Trace/ETEN` class to support tracing of packets dropped by the error module. The `handle` instance procedure to `Trace/ETEN` gets called once for every time a corrupted packet is dropped. The TCP sender has a choice of response methods, as discussed below.

We insert the error module in the link *after* the delay element in order to accurately model the fact that corruption is detected and the notification is sent by the node at the *end* of the link. We use the `ErrorModel/Uniform` Class to generate byte errors with a uniform random distribution. Any packet that contains one or more corrupted bytes is dropped and a selected ETEN mechanism is triggered. Alternative error models such as burst errors and channel fades are subject for future research.

We have modified the `Agent/TCP/FullTcp` and `Agent/TCP/FullTcp/Sack` to include two new commands: `eten-retransmit` and `fast-retransmit`. These TCP agents can now be commanded to retransmit a given sequence number regardless of TCP's current state. Using `fast-retransmit` causes the congestion window to be reduced as with Reno fast retransmit; using `eten-retransmit` does not alter the congestion control state, but does retransmit a segment. In the case of SACK, we use the pipe algorithm [5] for loss recovery and congestion control. As part of our work, we have fixed several bugs in the `Agent/TCP/FullTcp/Sack` implementation and contributed these to the ns-2 maintainers.

Two types of per-packet ETEN mechanisms are currently implemented – *Oracle ETEN* and *Backward ETEN (BETEN)*. As discussed earlier, both schemes assume that sufficient knowledge can be derived from the headers of the corrupted packets. Forward ETEN simulations have been left as future work, but we expect that FETEN will not perform as well as BETEN as the delay increases due to the increased signaling time.

The Oracle ETEN mechanism (as implemented in the simulator) instantaneously informs the TCP sending host of corrupted packets by directly invoking the sender's retransmit procedure.

The BETEN mechanism is implemented within the `Agent/Message/ETEN` Class. On each drop due to corruption, the agent sends an explicit notification message to the sending host. This message will suffer queuing, switching, transmission, and propagation delays incurred in the path from the notifying router to the sending host. The message is also susceptible to losses due to congestion or corruption. Different IP flows[5] can register with the `Agent/Message/ETEN` instance in each intermediate router from which they desire to receive ETEN messages.

We implemented a CETEN mechanism that is described in detail in Section 4. CETEN-specific software modifications include: (i) addition of fields and access methods to carry corruption and congestion survival estimates in the `Common` packet header (`struct hdr_cmn`); (ii) addition of variables and methods to the `ErrorModel class` to track packet corruption statistics and modify packet headers; (iii) addition of variables and methods to the `QueueMonitor class` to track congestion statistics and modify packet headers; and (iv) modification of `RenoFullTcpAgent class` to initialize the CETEN packet header fields and to

[5] Only TCP flows are of interest to these simulations.

decide if a packet loss was due to corruption. These CETEN modifications do not apply to the per-packet simulations described in this section.

We ported and incorporated the TCP Westwood code available from [24]. OTcl bindings to various ETEN parameters have been added to allow for configuration and control from the simulation scripts.

Simulation scripts (`csh` and `Tcl`) were written to set up the topologies, error models, ETEN mechanisms, the TCP flows under study, and the cross traffic. Additional scripts were written to process and plot the results of the simulations. The simulation code is available to the research community [4].

3.3 Baseline – no cross traffic over a single-hop topology (simulation 1)

In the first set of simulations, we investigate a single TCP flow over a single link with channel errors that result in packet corruption. In this set of simulations, there is no cross-traffic competing with the TCP flow. Examining ETEN in isolation provides an empirical upper bound on the gain in TCP goodput that is achievable using ETEN mechanisms.

The baseline for the simulations is the performance of TCP Reno and SACK under various error rates. We consider two near-ideal conditions for the error detection and notification:

1. Oracle ETEN – complete knowledge of the corrupted packet and instantaneous notification to the source.
2. BETEN – complete knowledge of the corrupted packet with real BETEN messages propagating back to the source.

Each simulation consists of a bulk TCP flow (FTP application) of 120 seconds duration with unlimited data to send. A variety of link delays and bandwidths representative of terrestrial wireline, wireless, and satellite link technologies are considered. The simulation parameter ranges are listed in Table 1. We consider a range of bit error rates from negligible (1.56×10^{-10}) to high (1.56×10^{-5}). In practice, high bit error rates are often lowered by using link-level FEC. However, we consider these high error rates for two reasons:

- We wish to determine how far we can push ETEN (at least in simulations) before the TCP goodput drops to an unusably low value.
- There are special cases in which end-to-end FEC is expensive in terms of bandwidth or other considerations and at the same time, implementing a link-layer FEC is not feasible. For example, software upgrades may be logistically difficult with satellite/spacecraft links.

We assume that the protocol processing delay in the router and end nodes is not significant. Therefore, the RTT for this set of simulations is twice the propagation delay on the link. The maximum segment size (MSS) of 536 bytes corresponds to a 576 byte MTU, the minimum that IPv4 routers must support. The router buffer size of 50 packets and TCP receive window size of 20 segments are ns-2 simulator defaults.

A set of plots that cover the entire parameter space simulated is included in Appendix A. The plots show the goodput using TCP Reno and TCP SACK with and without ETEN at various error rates in both directions. We have chosen three plots that represent a long thin network (LTN), a long fat network (LFN), and a short fat network (SFN) to illustrate the performance of ETEN.

Figure 4 shows that Oracle ETEN provides a significant improvement in goodput, particularly at high bit error rates of 1.56×10^{-6} to 1.56×10^{-5}. For example, at a bit error rate (BER) of 1.56×10^{-5} over the LFN link

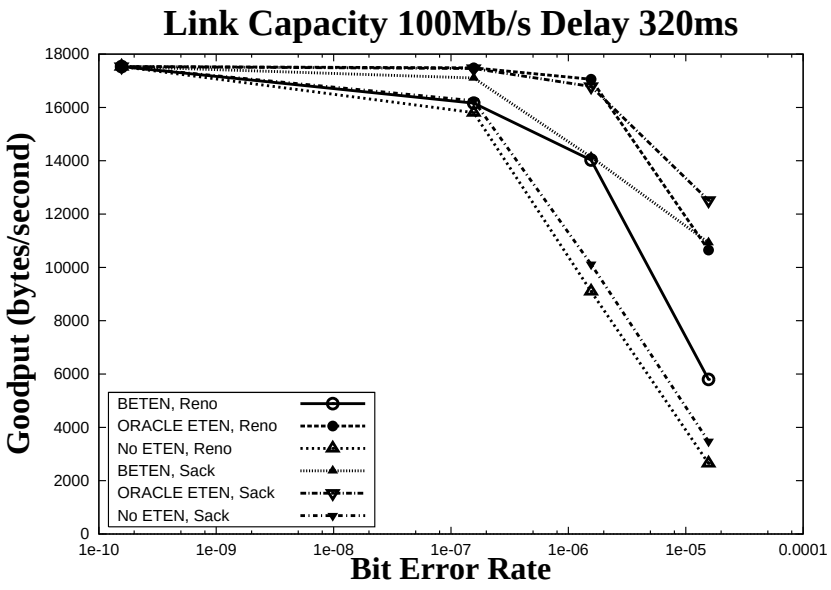


Figure 4: TCP with ETEN over an uncongested long fat network (LFN)

of 100 Mb/s capacity and a 320 ms one-way delay, the goodput using SACK with Oracle ETEN is about 3.5 times the baseline goodput when using the stock SACK variant. The goodput using Reno with Oracle ETEN is almost four times the baseline goodput using Reno alone. The goodput using BETEN with SACK is more than three times the SACK baseline, and the goodput using BETEN with Reno is about 2.4 times the Reno baseline. The figure also illustrates that when the errors are not as prevalent on the link (e.g., 1.56×10^{-7}) the ETEN mechanisms have a relatively small impact because errors have only a small impact on stock TCP.

For networks with other link bandwidth $\times$ delay products, much higher goodput improvements are shown. For instance, the SFN link case is shown in Figure 5. SACK with BETEN provides 7 times the goodput of baseline SACK for a 100 Mb/s, 10 ms link at a BER of 1.56×10^{-5}. As expected for SFN, the performance using BETEN with SACK is close to that of Oracle ETEN at low error rates. However as the BER increases, the chances of losing a notification also increases and we see that gains from BETEN begin to diminish. Using BETEN with SACK outperforms BETEN with Reno; this may be because the ability of SACK to correct multiple losses complements ETEN. As with the LFN case described above, we note that no ETEN mechanism helps connections experiencing low BERs.

Finally, Figure 6 shows ETEN performance over a long-thin network. The results are similar to the LFN case indicating that the goodput of the stock TCP (as well as the improvements from Oracle ETEN and BETEN) is dominated by the delay.

From the simple simulations presented in this section we can derive several conclusions:

- We have confirmed previous studies that show TCP performance degradation in the presence of high BER links.
- In general, TCP SACK performs better than TCP Reno due to the ability of TCP SACK to mostly decouple loss recovery from congestion control. The goal of ETEN is to decouple recovery of lost packets from congestion control decisions. These goals are complimentary as evidenced by the fact that the BETEN+SACK TCP variant outperforms the BETEN+Reno TCP variant.

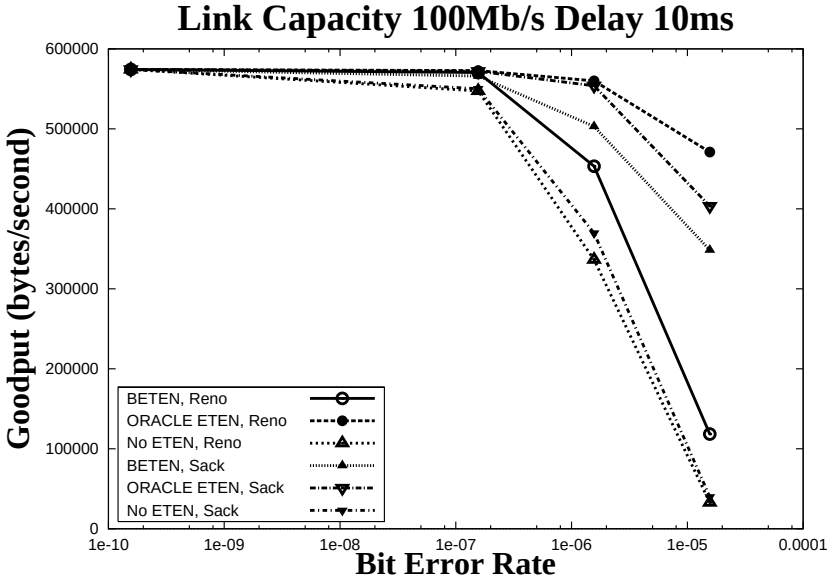


Figure 5: TCP with ETEN over an uncongested short fat network (SFN)

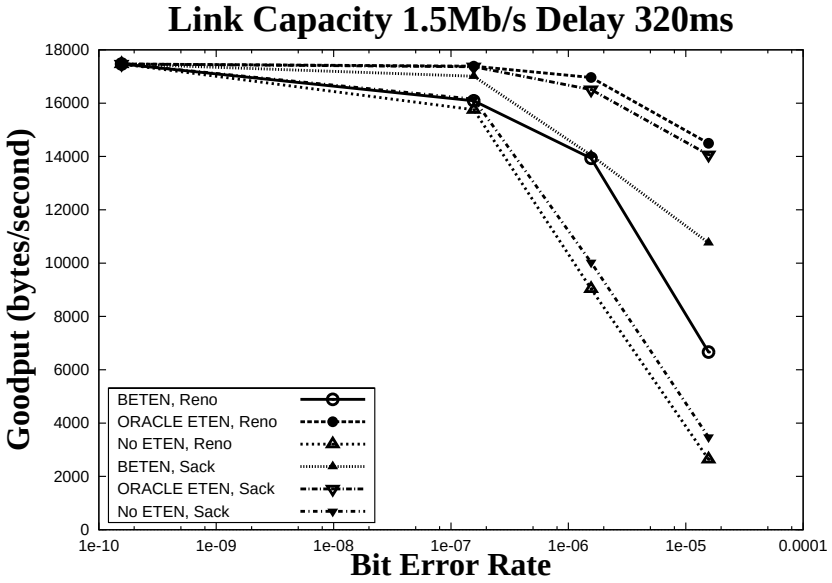


Figure 6: TCP with ETEN over an uncongested long thin network (LTN)

- Both Oracle ETEN and BETEN show performance improvements when compared to stock TCP. The performance is increased by a factor of 7 in some cases with the BETEN+SACK TCP variant.
- Oracle ETEN is effective in keeping TCP goodput high as BERs increase. We only observe significant goodput degradation when the BER approaches $O(10^{-5})$.
- As expected, BETEN improves the goodput less when compared to Oracle ETEN (although, the improvement is still often significant).

3.4 Multi-hop topology with no cross-traffic (simulation 2)

In the second set of simulations we investigate a single TCP flow, but over a 3-hop linear topology of identical links. There is no cross-traffic and the same set of parameter ranges are used as in the first set of simulations. In this case, packets can be corrupted in any intermediate link, and therefore notifications can come from any intermediate router or end-system. The plots for a 3-hop topology are included in Appendix B. We find that the conclusions outlined above for the single hop topology continue to hold when traversing a 3-hop linear topology. Specifically, we observe that:

- Oracle ETEN and BETEN always improve performance when compared to stock TCP.
- Oracle ETEN (the ideal case) is able to keep goodput high in situations where stock TCP performance degrades substantially. This indicates that differentiating between congestion-based loss and corruption-based loss shows the potential for large performance gains.
- BETEN performance approaches the ideal as the message notification delay decreases provided the probability of losing the notification is small.

3.5 Multi-hop topology with UDP cross traffic (simulation 3)

In the third set of simulations, we included UDP cross traffic. We investigate a single TCP flow with and without BETEN. The cross traffic generators send UDP traffic across the middle hop. They are started 50 ms after the TCP flow. The traffic generators use an exponential on-off model with mean burst and mean idle times, both set to 500 ms. The packet generation rate for each flow is set to one-fourth of the link bandwidth. The first competing UDP flow is started 40 ms after the TCP flow of interest with the subsequent flows started at 50 ms intervals thereafter. We varied the number of flows from 0 – 16. However, in this report we only present the results of simulations involving 0, 4 or 8 UDP flows. The remaining points in the parameter space offer similar results. Each data point in the plots is from a single run. The UDP cross-traffic can cause congestion in the routers connected to the middle hop (depending on the number of UDP flows employed in a particular simulation). For the results presented in this report the cross-traffic load can be non-existent, fully utilize the bottleneck capacity[6] or twice the bottleneck capacity. We use the same parameter space as in simulation 2.

The performance of SACK TCP with BETEN in the case of a 3-hop LTN under congestion from UDP cross-traffic is illustrated in Figure 7. BETEN continues to improve goodput at high error rates, even in the presence of competing UDP traffic. Additional plots from these simulations are included in Appendix C. From this set of simulations, we derive the following conclusions:

[6]Even in the cases where the UDP flows do not fully use the bottleneck bandwidth we see congestion losses due to the flow-of-interest's use of the bottleneck link.

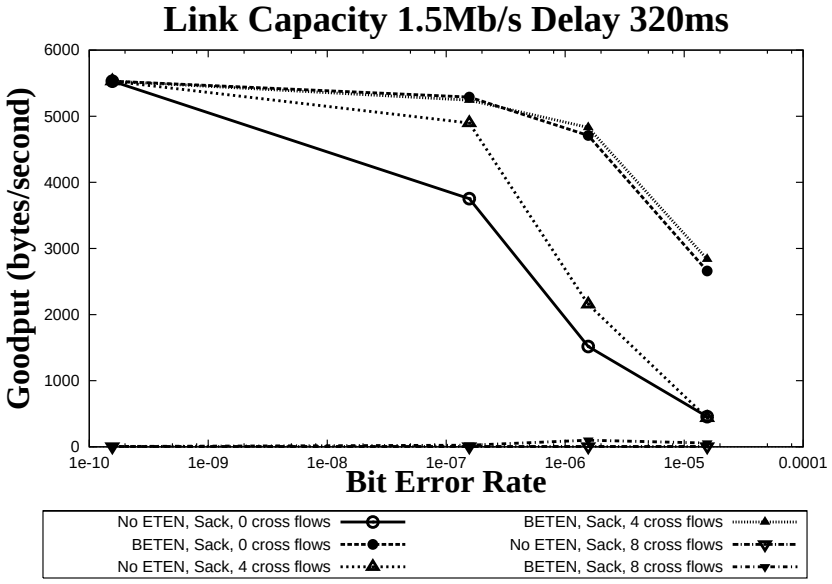


Figure 7: SACK TCP with BETEN over a congested 3-hop LTN of one way path delay 960 ms with UDP cross-traffic

- At high BERs and with low congestion both ETEN schemes offer increases in goodput when compared to stock TCP. This result is in agreement with the previous two subsections in which no cross-traffic (i.e., no congestion) was employed.
- As congestion increases BETEN allows TCP to attain goodput that more closely resembles the error-free case than stock TCP, especially as the link quality degrades.

3.6 Multi-hop topology with competing TCP flows (simulation 4)

In the fourth set of simulations, we investigate a single TCP flow over a 3-hop linear topology and background TCP traffic. ETEN messages may be generated by any intermediate router or end system. In this set of simulations, the middle hop experiences cross traffic in both directions in the form of TCP flows that do not use ETEN. The first competing flow is started 40 ms after the TCP flow of interest with the subsequent flows started at 50 ms intervals thereafter. The competing flows use the BSD Tahoe variant of TCP with the ns-2 default setup. Once started the competing TCP flows send data continuously for the remainder of the simulation. The links experience errors in both directions and due to the cross-traffic, we observe congestion-based losses in this simulations. We have simulated $0 - 16$ competing TCP flows in each direction. However, in this report we show only the results from using 0, 4 or 8 competing connections for brevity (the remaining results are consistent with those outlined in this report).

The performance of SACK TCP with BETEN in the case of a 3-hop LTN under congestion from TCP cross-traffic is illustrated in Figure 8. We only plot the goodput of the flow of interest that spans the 3-hops (and not the cross-traffic flows that span a shorter path). As the cross-traffic increases from 0 to 4 to 8 flows, we note that the flow-of-interest's share of the bottleneck link is reduced and therefore the goodput shown in the plot decreases. As in the last section, BETEN improves performance at high BERs, even as congestion on the bottleneck link increases. We note that when 4 competing flows are active, BETEN allows the flow of interest to roughly maintain its share of the bottleneck link across all BERs shown in the figure. In addition,

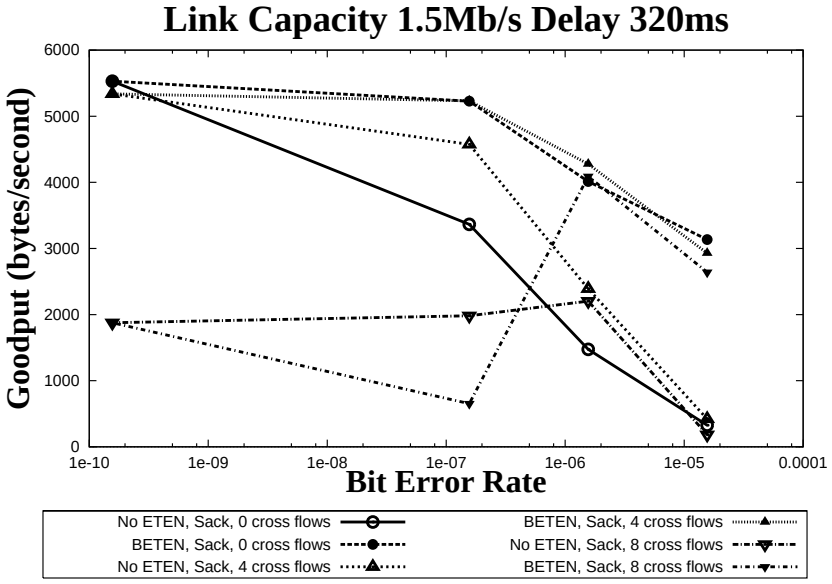


Figure 8: SACK TCP with ETEN over a congested 3-hop LTN of one way path delay 960 ms with TCP cross-traffic

when 8 competing flows are enabled BETEN allows the flow of interest to *increase its share of the bottleneck resources*. This is explained by the errors being spread across more TCP connections. That is, assuming the BER is constant, as the number of competing connections increases the likelihood of the flow of interest being affected by a particular error is reduced (because its share of the network decreases). In addition, the competing traffic in our simulation did not use any ETEN mechanism. Therefore, when experiencing an error the cross traffic needlessly reduces the load being placed on the network – and, hence allowing the flow of interest to use more of the bottleneck bandwidth.

In general, across a wide range of delays and bandwidths, we found that BETEN improves TCP goodput over the baseline case as the BER and level of congestion increases. See Appendix D for the complete set of plots from this set of simulations.

3.7 Comparison of ETEN to TCP Westwood

Thus far we have evaluated the performance of ETEN with TCP Reno and TCP SACK. Reno and SACK use an additive-increase multiplicative-decrease (AIMD) congestion control strategy in which the congestion window is halved when a loss (congestion) is inferred and the window grows linearly otherwise. Recently proposed modifications to TCP congestion avoidance include using bandwidth estimation techniques. TCP Westwood [12] is a representative congestion avoidance strategy based on bandwidth estimation. TCP Westwood has been shown to perform well under high error rates in simulated comparisons to TCP Reno and SACK TCP. Here, we compare via simulations the performance of ETEN with Reno and SACK[7] against TCP Westwood. We simulate over a wider range of error rates and congestion levels than the previous studies on TCP Westwood.

[7]The Reno simulations are not discussed in this report, however the trends observed in the Reno runs are similar to those shown in the SACK runs. We present the SACK variant of TCP as it shows the best performance in the simulations presented in the previous sections.

3.7.1 Comparing ETEN to TCP Westwood with no cross-traffic (simulation 5)

In this simulation, we compare the performance of TCP Westwood to Oracle ETEN and BETEN mechanisms over the range of bandwidths, latencies, and packet corruption rates (listed in Table 1). We ported the TCP Westwood code changes from a previous version [24] to ns-2 version 2.1b7a.

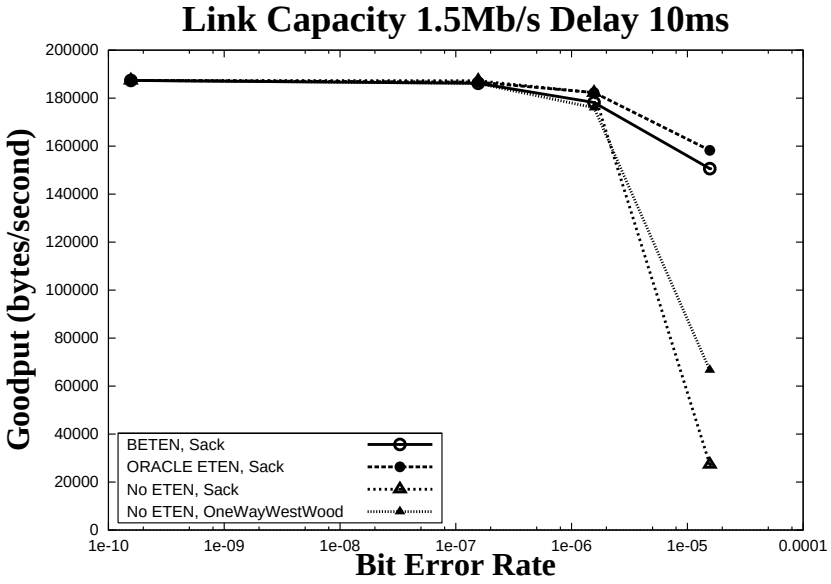


Figure 9: TCP Westwood versus SACK TCP with ETEN

The relative performance of TCP Westwood in comparison to SACK TCP (stock, with Oracle ETEN, or with BETEN) in the absence of congestion is illustrated in Figure 9. The figure shows the results from simulations involving a 1.5 Mb/s link with a one-way delay of 10 ms. Additional plots from these simulations are given in Appendix E. As expected, Oracle ETEN performs the best. TCP Westwood performs similarly to BETEN except when the BER reaches the 10^{-5} range. However, when the error rate is 10^{-5} BETEN outperforms TCP Westwood – by a factor of 2 in some of our simulations. Also, we note that TCP Westwood performs no better than stock SACK TCP when the BER is relatively low.

3.7.2 Comparing ETEN to TCP Westwood with UDP cross-traffic (simulation 6)

In this set of simulations, we compare the performance of TCP Westwood when both congestion and corruption losses are present. Figure 10 shows the performance of TCP Westwood and BETEN over a 3-hop linear topology with 1.5 Mb/s links each with a one-way delay of 10 ms. We use competing UDP traffic for these simulations, as outlined in section 3.5. The plot shows that BETEN with SACK outperforms TCP Westwood at a BER of 10^{-5} regardless of congestion level. In addition, the plot shows that when no competing flows are present TCP SACK performs similar to or better than TCP Westwood.

The plot shows that when using 4 competing flows TCP Westwood outperforms stock SACK TCP (except at a BER of 10^{-5}). One data point of particular interest in this plot shows Westwood performing 6 times better than TCP SACK in the error-free case. This indicates that TCP Westwood may be more aggressive than TCP. This phenomenon is out of scope for the current work, but a deeper study into this behavior would be useful future work.

Additional plots that demonstrate a similar behavior over a range of bandwidths and delays are included in Appendix F. At high error rates and moderate congestion, BETEN's ability to distinguish between corruption and congestion losses provides performance improvements over the TCP Westwood strategy that relies on intelligent bandwidth estimation alone. The Westwood strategy, however, shows an advantage under heavy congestion (4 competing flows) with low to moderate error rates.

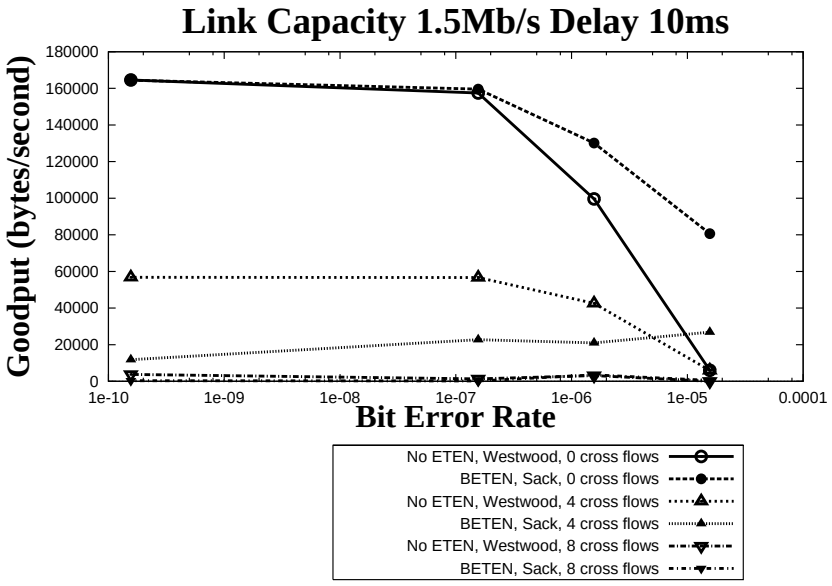


Figure 10: TCP Westwood versus SACK TCP with ETEN

4 Cumulative ETEN Strategies and Performance

In Section 2.2 we described cumulative transport error notification techniques that are applicable when sufficient information about specific packet drops is not available to the transport layer endpoint. Rather, using cumulative ETEN (CETEN) the TCP sender relies on packet drop statistics to drive the behavior of the congestion control algorithms. In this section, we describe one CETEN strategy that we study in detail.

4.1 Transporting corruption survival probability estimates

The mechanism we employ in our study adds two survival-probability fields to each packet. These fields indicate the probability that a packet avoids corruption – one field for each direction. In the ns-2 implementation, these fields are included in the `Common` packet header. The forward path survival probability field is initialized to 1 by the source of the packet and is updated by routers along the path. Therefore, when a packet arrives at the receiver the survival probability contained in the packet is the survival probability of the path. The transport endpoint at the destination keeps a record of the survival probability of the forward path. When the endpoint generates a packet or an ACK it copies this saved value into the reverse path survival probability, and initializes the forward path survival probability to 1.

An alternative method of transmitting information about survival probabilities could use something akin to the ICMP Traceback model [39]. Using such a mechanism each router would probabilistically transmit a message to the sender of an incoming packet informing the sender of the error rate on the incoming link. After awhile

the sender will have received information from each router along the path and can calculate the current path error rate. This mechanism has the advantage of not requiring additional header space in each packet. However, it also has the disadvantage of adding traffic to the network and requiring the sender to transmit a fairly large amount of traffic to ensure enough path information is collected. We leave investigation of this mechanism as future work.

4.2 Computing the per-link packet corruption survival probability at each router per-link

Each router multiplies the forward path survival probability field in each packet by the router's own estimate of the survival probability for the link on which the packet arrived. The estimate of the survival probability for the link is computed on receiving each packet on the link as follows:

$$p_i = \alpha x_i + (1 - \alpha) p_{i-1} \tag{1}$$

where $x_i = 1$ if the i^{th} packet is received correctly, and $x_i = 0$ if the i^{th} packet is corrupted. The parameter α is chosen from $[0, 1]$. We use $\alpha = 0.5$ in our experiments and leave examining the impact the exact value of α has on performance as future work.

4.3 Transport endpoint strategies to use CETEN to discern congestion from corruption

In this section, we address the question of what the sender should do with the corruption probability estimates and how TCP's algorithms may be changed to incorporate this new information.

First, given that TCP has inferred loss(es) from duplicate acknowledgments (dupacks), selective acknowledgments (SACKs) and/or timeouts, we need a way to decide whether to invoke congestion control procedures. For example, a probabilistic decision function can generate a random number in $[0, 1]$ and compare it to the conditional probability that given a loss has occurred, it is due to corruption. If, probabilistically, a particular loss is attributed to packet corruption the lost segment can be retransmitted without modifying the sender's congestion control state. However, in order to make this mechanism work the TCP must be able to estimate this conditional probability.

Loss can be either due to congestion or corruption. In theory, if a TCP knew how to ascertain the fraction of losses due to one cause (say, losses due to corruption) and if the TCP can determine the total losses, then the TCP can determine the losses due to the other cause. The problem with this scheme is that determining the total loss is not always easy in the case of TCP. The preciseness of loss detection in TCP depends on the particular variant being used. For instance, loss inferred during the slow start phase that follows the expiration of the RTO timer often ends up retransmitting segments that have already arrived at the receiver (see [5] for an example of this). On the other hand, if SACK is used to recover all losses without reverting to the RTO timer the loss detection is precise [5] and TCP may come up with an accurate estimate of the total drop rate. For the purposes of CETEN it is not clear how precise the estimate of total loss would have to be and determining this is beyond the scope of the current work. However, this is an area that should be explored in the future.

The alternative approach is to have the network inform TCP about the current congestion-survival probability, much like the scheme outlined above for corruption information. The biggest weakness of such an "in-the-network" scheme is that if some congested routers do not participate they cause the sender to overestimate the number of losses attributed to corruption and therefore inject more traffic into the network than appropriate. In-the-network strategies require less accounting on the part of the TCP sender/receiver; however, there is a non-zero probability of misdetecting a congestion loss as a corruption loss. There is an associated security/reliability

risk that a misbehaving router may adjust the value of the corruption survival probability to a low value to cause the senders to retransmit without backing off, thereby leading to congestion.

For the work outlined in this report we chose an in-the-network scheme due to the accuracy of the mechanism. We have added an additional pair of packet header fields to carry the *congestion* survival probability, computed in a way similar to the corruption survival probability outlined above. Every intermediate node estimates the packet survival probability from congestion (in addition to the probability of surviving corruption). This information gets recorded in subsequent packets traversing the router.

At the transport endpoint, on a loss event, we examine the last observed estimates of congestion survival p_{cong} and corruption survival p_{corr}. The loss event is taken as as a congestion event with probability $\frac{p_{\text{cong}}}{p_{\text{cong}}+p_{\text{corr}}}$ and the standard adjustments are made to the congestion control state. If the loss is declared a corruption loss, a retransmission occurs and congestion control state is not altered.

If the strategy in our simulations is to be realized in practice, every router will be required to maintain per-link congestion and corruption statistics and pass them via packets in ETEN-capable flows. While this is implementable (using statistics already being collected by modern routers), universal deployment of CETEN-capable routers presents practical challenges. Furthermore, how accurate and recent the end-to-end estimate (of corruption and congestion survival probabilities) will be (and needs to be) requires real world implementation and experimentation.

4.4 Cumulative ETEN performance with UDP cross traffic (simulation 7)

Figure 11 illustrates the goodput of a 120 second FTP session over a 3-hop topology of 100 Mb/s links each with a one-way delay 100 ms in the presence of UDP cross traffic. The bit error rate of each link in the topology is varied in the range 1.56×10^{-10} to 1.56×10^{-5}. The link in the middle has cross traffic in the form of 0, 4, or 8 UDP flows in each direction. The traffic generators use an exponential on-off model with mean burst and mean idle times, set to 500 ms. The packet generation rate for each flow is set to one-fourth of the link bandwidth. The first competing UDP flow is started 40 ms after the TCP flow of interest with the subsequent flows started at 50 ms intervals thereafter.

The figure shows that under some heavy BERs (e.g., 10^{-6}) CETEN with TCP Reno performs better than standard TCP Reno. However, at the highest BER tested (1.56×10^{-5}) CETEN offers roughly the same performance as stock TCP Reno under all congestion levels. Further, under low BERs CETEN performance is roughly the same as stock TCP Reno, except under heavy congestion (4 UDP flows), in which case CETEN with TCP Reno performs slightly worse than TCP Reno alone. Appendix G shows CETEN behavior for the parameter space investigated in this report. From these plots we note that CETEN performance is generally roughly the same as TCP Reno when the BER is low (e.g., 1.56×10^{-7}) or high (1.56×10^{-5}). When the error rates are modest CETEN provides performance benefits when compared to TCP Reno. Finally, CETEN offers greater performance benefits when there is less congestion.

4.5 Cumulative ETEN performance with TCP cross traffic (simulation 8)

Figure 12 illustrates the goodput of a 120 second FTP session over a 3-hop topology of 100 Mb/s links each with a one-way delay 100 ms in the presence of TCP cross traffic. The bit error rates of the link are varied in the range 1.56×10^{-10} to 1.56×10^{-5}. The link in the middle has cross traffic in the form of 0, 4, or 8 TCP flows in each direction. The first competing flow is started 40 ms after the TCP flow of interest with the subsequent flows started at 50 ms intervals thereafter. The competing flows use the BSD Tahoe variant of TCP with the ns-2 default setup.

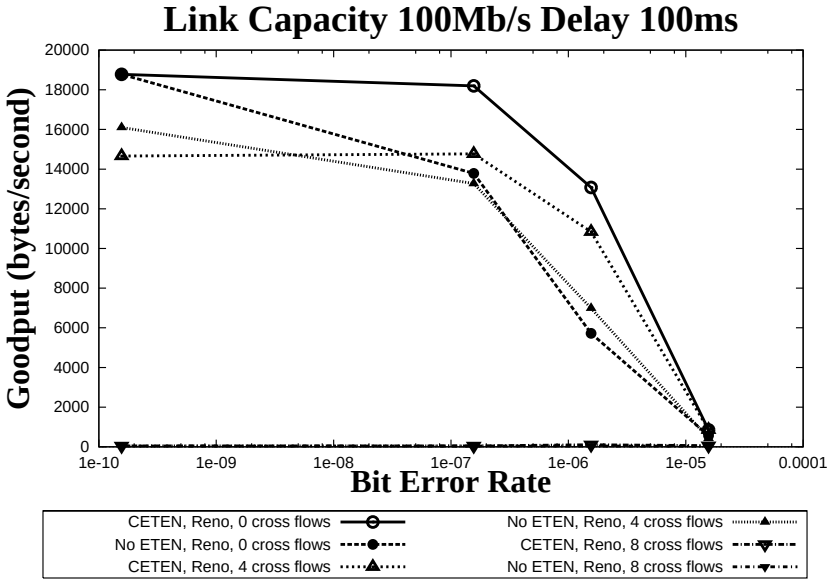


Figure 11: CETEN Performance with TCP Reno and UDP cross traffic

We notice that under all congestion levels, CETEN offers moderate goodput gains over TCP Reno, except at high BER (1.56×10^{-5}). We notice that the goodputs using CETEN with Reno at a BER of 1.56×10^{-6} with 8 competing flows is higher than with only 4 competing flows. This is explained by the errors being spread across more TCP connections. That is, assuming the BER is constant, as the number of competing connections increases the likelihood of the flow of interest being affected by a particular error is reduced. In addition, the competing traffic in our simulation did not use any ETEN mechanism and they needlessly reduce the transmission rate when they experience corruption losses. This allows the flow of interest to use more of the bottleneck bandwidth. Plots for the entire range of the simulation are included in Appendix G. At some data points shown CETEN provides noticeably higher performance than TCP Reno. However, in the majority of the cases with high BERs, CETEN does not provide significant gains in performance.

4.6 Discussion of CETEN simulations

The CETEN simulations we conducted as part of this investigation show CETEN to be a promising approach in some situations. In other situations CETEN offers worse performance than TCP Reno. We feel that further investigation into additional CETEN mechanisms is warranted before making conclusions on the feasibility of CETEN in general. For instance, an investigation into how well the end system can estimate the total loss rate and use that for determining the fraction of losses caused by congestion may shed additional light on CETEN (and make it more feasible to deploy).

5 Combining CETEN Strategies with Dynamic FEC Adjustment

There are fundamentally two ways in which FEC strategies can be combined with CETEN: either the error correction code can be contained entirely within each packet or it can be distributed across multiple packets. In the first case, each packet can include additional bits of error correcting information, and each intermediate router must detect and if possible correct errors before forwarding the packet. A large number of error correcting

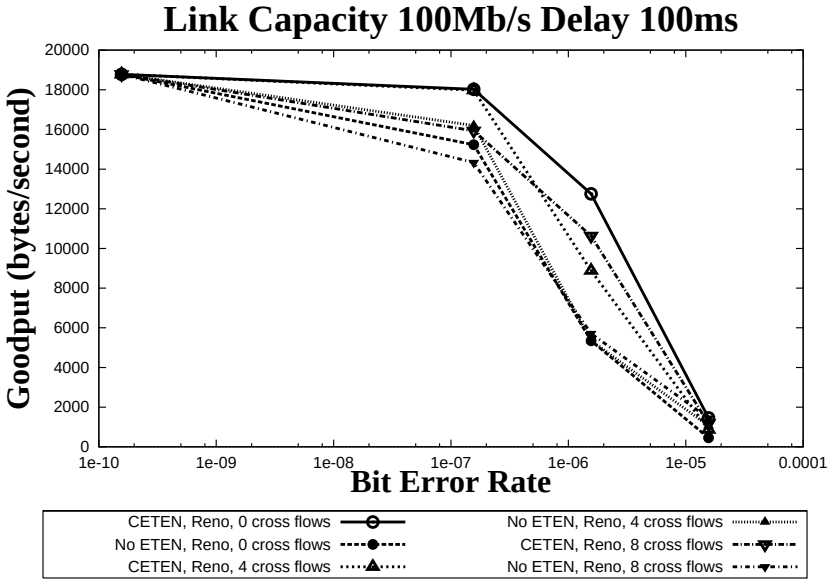


Figure 12: CETEN Performance with TCP Reno and TCP cross traffic

codes that are effective under different error models are available.

In the second case, erasure codes can be used that allow corrupted packets to be dropped while allowing the end points to recover the information from additional redundant packets. The Stutter XOR scheme [11] is an example of a simple erasure code. More sophisticated codes have been applied to packet switched networks [21] [13] [16].

In this section we will discuss the issues and challenges in combining CETEN strategies with both kinds of FEC strategies mentioned above.

Traditionally, a given error correcting code has been used to protect against a known error rate and model. In a heterogeneous wireless environment (that may include, for example, both wireless access links and long haul satellite links), there may be dynamic variance in the channel characteristics (such as available bandwidth) and error model (for example, bit, burst, or fade). A static coding scheme must balance the processing and bandwidth cost against protection against a particular error rate and model. While adaptive links are possible, there is an opportunity to use end-to-end mechanisms to track the cumulative error rates along the path and dynamically increase or decrease the strength of end-to-end FEC. End-to-end FEC can be used even when per link coding is not available, or easily replaceable (e.g. for satellite links).

However, there are significant challenges to be considered in combining FEC and CETEN. Any reliable transport protocol must still provide end-to-end ARQ to guarantee packet delivery. TCP, in particular, uses ARQ in its combined error, flow, and congestion control algorithms; the addition of, and interaction with, FEC may add significant protocol complexity.

In the case of satellite or wireless links, per packet FEC cannot protect against all non-congestion packet losses, for example, channel fades. Furthermore, IP routers simply drop erroneous packets to prevent mis-forwarding. With per packet FEC, intermediate routers must correct packet headers (provided there is no IP-IP encapsulation, else the payload may also have to be corrected at intermediate routers) to ensure mis-forwarding. Furthermore, for any given path MTU, the use of variable strength FEC means that the MSS seen by TCP will fluctuate with the error rate.

The interactions of end-to-end TCP mechanisms for flow control, error recovery, and congestion avoidance with erasure codes is much more subtle. There is this tension between erasure codes, on the one hand, trying to mask all packet losses (whether due to congestion, corruption, or fading) and prevent retransmissions, while TCP on the other hand, relying on the congestion losses as control feedback to congestion avoidance mechanisms. This masking of losses challenges the fundamental ETEN goal of being able to discriminate between corruption and congestion packet losses.

TCP ACKs carry the sequence number of the next byte of data the receiver expects to arrive. This allows the sender to determine packet losses and adjust the congestion window. When erasure codes are used, this feedback is insufficient since the last segment being accounted for (as received) may belie the fact that some packets could have been lost due to congestion (but were reconstructed at the receiver). Enough packets must be dropped so as to exceed the error capability of the code before the TCP sender is actually notified of congestion. This added delay might make the congestion avoidance loop unstable.

Solving this problem requires that we keep track not only the sequence number of the payload data but also the sequence number of the encoded packets. In this case, TCP congestion avoidance should use this latter sequence number. This will require the addition of this information to the IP or TCP packet headers (perhaps in the form of an option).

Furthermore, with erasure codes, the receiving TCP has to wait for the possibility of subsequent packets correcting a loss. This can conflict with the settings of the retransmit timer and the delayed acknowledgment timer.

Implementation of FEC mechanisms in ns-2 with the corresponding analysis of the practical issues involved in using such a mechanism is recommended for a future phase of this work.

6 Mechanisms to Detect Corrupted Packets

Mechanisms are needed in intermediate routers and in end-systems to detect corrupted packets so that explicit transport error notification can be performed. Some of these mechanisms can be implemented without changing current protocol specifications, but some others (such as marking an IP packet on checksum failure and not dropping it) require changes. Furthermore, the layer in which the error is detected (e.g., link or network) must co-operate with the layer that is performing recovery (namely, the transport layer in the case of ETEN).

6.1 Link-layer mechanisms

Many corrupted packets are not received at layer 3, since layer 2 mechanisms generally do not forward corrupted layer 2 packets to the higher layers. Therefore, modification of current link-layer mechanisms will play a major role in making ETEN possible.

A number of link-layers already provide link-level error detection (and correction) mechanisms. An example of this is the 32-bit Frame Check Sequence (FCS) field in the 802.11b MAC header. Frame errors observed at the link layer can also be used to keep cumulative statistics of errors and inform flows that use the link.

In the case of CETEN, the link layer must collect error statistics and make them available to the higher layers. In the case of FETEN or BETEN, the link layer must co-operate with the higher layers to determine relevant flow-specific information needed for generating an ETEN message.

6.2 IP checksum

The IP checksum field is a 16-bit one's complement sum computed on all 16-bit words in the IP header only. Since some header fields change (e.g., time to live), the checksum is recomputed and verified at each point that the IP header is processed. For purposes of computing the checksum, the value of the checksum field is zero [25]. According to RFC 1812 on IPv4 router requirements [28]:

> A router MUST verify the IP checksum of any packet that is received, and MUST discard messages containing invalid checksums. The router MUST NOT provide a means to disable this checksum verification. A router MAY use incremental IP header checksum updating when the only change to the IP header is the time to live. This will reduce the possibility of undetected corruption of the IP header by the router.

Since the one's complement checksum does not provide any error correcting capability, the router or end-system receiving an IP packet with an incorrect checksum may not correctly associate the packet with its source. However, this information can be used to keep cumulative error statistics on the incoming link and this information can be provided to all flows using the link.

An alternative is to send a packet-specific notification to the source IP address determined from the corrupted packet and then leave each node responsible for discarding false notifications. However, a misbehaving router can cause an unrelated host to receive false notifications which that host will have to process.

6.3 TCP checksum

The 16-bit checksum field in the TCP header is the 16-bit one's complement sum of all 16-bit words in the TCP header, the packet data, and a 96-bit pseudo-header conceptually prefixed to the TCP header. The pseudo-header contains the source and destination IP addresses, the protocol, and TCP length (data length plus TCP header length). If a segment contains an odd number of octets to be checksummed, the last octet is padded on the right with zeroes to form a 16-bit word for computing the checksum, but the pad itself is not transmitted. The pseudo-header provides some protection against misrouted packets. For purposes of computing the checksum, the value of the checksum field is zero.

Unfortunately, TCP does not provide separate checksums for header and data, and includes portions of the IP header. If the TCP checksum is incorrect but the IP checksum is correct, a notification to the sending host becomes possible, but the source and destination TCP ports and the TCP sequence number cannot be ascertained with guaranteed accuracy.

On the other hand, if IP checksum is incorrect but the TCP checksum is correct, then the source and destination IP addresses (as well as the TCP ports, and the TCP sequence number) can be assumed to be correct. The IP header must have been corrupted in some other field. To determine this case, we require inter-layer co-operation between the IP and TCP layers.

6.4 IPsec

In the case of IP Encapsulating Security Payload (ESP) [31], if the encrypted ESP packet is corrupted, it can be detected if authentication is selected for the security association. However this can be done only at the endpoint of the security association. Thus IPsec renders any sort of per-packet scheme useless at the level of a TCP connection. We discuss this in more detail in the next section.

7 ETEN Security Issues and Implications

In this section we introduce security issues related to ETEN mechanisms, as well as potential solutions that can be the basis for a detailed study in this area. We consider two kinds of issues:

1. New opportunities to exploit security vulnerabilities due to ETEN mechanisms, such as:
   - denial of service attacks
   - spoofed ETEN messages can cause the TCP sender to be aggressive, thereby causing congestion
2. Issues involving interoperability with existing security mechanisms such as encrypted tunnels

7.1 Potential security vulnerabilities due to ETEN

ETEN techniques (such as BETEN, for example) that require out-of-band messages are vulnerable to distributed denial of service (DDOS) attacks because networks that plan to use this form of ETEN will have to allow such messages to enter or leave their networks. This makes it possible for an adversary to launch a DOS attack by bombarding a host (or a network) with ETEN messages. This can minimally overwhelm the victim host, but if launched as a distributed denial of service attack from a large number of hosts (that have been compromised by an Internet worm, for instance), an attack can overwhelm the capacity of entire networks (similar to one described in detail in [8]).

DDOS vulnerabilities are not specific to ETEN, but rather they are due to the lack of a secure signaling protocol for the Internet (ICMP is not secure). However, since the response to ETEN is a retransmission, the effects are severe. ETEN attacks can cause congestion collapse of a network; in contrast, the response to ECN is a reduction in the transmission rate, and a spurious ECN message can at worst throttle the connection.

Denial of service attacks may come from disingenuous (or compromised) end-systems, from compromised intermediate nodes, or from nodes that masquerade (by IP spoofing, for example) either as intermediate nodes or end-systems corresponding to an existing TCP flow.

Generally, it is considered more difficult to compromise intermediate systems such as routers that are managed professionally. IP tracing techniques [20, 22, 39] can make it easier to track spoofing attacks, hence providing a disincentive for conducting such attacks. However, currently such mechanisms are not generally available.

If spoofing can be eliminated, then it may be possible to use stateful inspection to check that ETEN packets correspond to endpoint address/port/sequence information of existing flows to filter some attack packets. Note, however, that the problem of DDOS attacks coming from a large number of compromised end-systems is difficult and is not specific to ETEN. Nonetheless, we conclude that ETEN techniques that do not require additional messages are preferable from the security standpoint.

ETEN mechanisms may be vulnerable to another more subtle and indirect attack. A malicious adversary can send false notifications corresponding to packets that are either not dropped or were dropped due to congestion. This can induce the sender into retransmitting packets unnecessarily or into bypassing congestion avoidance and continue transmitting at a higher rate than appropriate for the given network conditions. This attack in isolation (on a single flow) can cause limited damage. However, if a coordinated attack were launched on many TCP flows on a heavily loaded network, the attack can potentially drive the network into congestion collapse (see for example, [7]). This form of attack requires that attackers obtain the necessary information (such as IP address, TCP port, and TCP sequence number, and packet drop/delivery information). The attackers may do so by snooping on the network, by initiating sessions from compromised end-systems, by hijacking TCP sessions, or by compromising routers.

A similar concern stems from misbehaving receivers [19] that attempt to gain an unfair share of the bandwidth (rather than attempt a denial of service per se). A misbehaving receiver could induce an ETEN-capable sender into never reducing the congestion window by informing the sender that all the losses are due to corruption. Hence, the sender will transmit at an inappropriately high rate.

7.2 Interoperability with existing security mechanisms

The use of encryption can prevent deep header inspection. For example, IPsec [30] hides TCP port information; IPsec tunnels also hide the original source address. This makes it difficult for intermediate routers to determine the correct TCP endpoints to which ETEN messages should be delivered. This problem also arises in other contexts such as NAT [38], ECN [37], or hierarchical stacking of MPLS labels [34], where deep header look-up may be computationally expensive, may require additional hardware, or may be prohibited.

The intermediate router may be able to use the packet sequence number and the security association information to forward ETEN messages to the tunnel endpoint, which in turn, may resolve the appropriate end system to deliver the ETEN message. This is an area of future work.

We note that the particular ETEN implementation should be analyzed for additional implementation-specific vulnerabilities. An ETEN prototype as part of future work is recommended to properly test and analyze vulnerabilities.

8 Conclusions and Future Work

In this study we propose explicit transport error notification mechanisms to improve the throughput of TCP over error-prone wireless and satellite links. We have implemented the proposed mechanisms under the ns-2 simulator and evaluated their performance via simulations over a wide range of bit error rates, link bandwidths, and traffic conditions. We have addressed practical deployment issues including security considerations.

To recapitulate, the primary technical objectives of this study include:

- determine if ETEN is a worthwhile technology to implement (determine what improvements in bandwidth utilization are possible);
- determine if ETEN is practical to implement;
- identify possible ways to implement ETEN;
- identify the potential pitfalls regarding ETEN such as issues regarding security, particularly, denial of service;
- identify the similarities of ECN and ETEN and the possible interactions of ECN with ETEN; and
- recommend additional research that must be performed in order for this technology to eventually be incorporated into the protocol standards.

In this concluding section we address these objectives. This effort was scoped to be an initial study based on simulations. Emulations in a testbed and live testing over real networks were considered out of scope of this effort. Our goal in this study was to perform a broad sweep of the parameter space rather than focus on a particular mechanism. The simulation code is available to the research community for further experimentation,

for example, to focus closely on a smaller region of the parameter space. Examining the data provided in the appendices may offer further insights for future work.

In the following subsections, we summarize the salient conclusions from this study; we list the presentations, reports, and software that were generated during this study; and finally, we present recommendations for future work in this area.

8.1 Salient conclusions

Per-packet ETEN mechanisms offer substantial gains in bulk TCP goodput in the absence of congestion. Our results indicate that per-packet ETEN mechanisms can improve the performance of TCP Reno and TCP SACK in uncongested networks across a wide range of path capacities and delays. About *seven times* improvement in goodput was observed in some cases in the BER range of 10^{-5} to 10^{-7}.

Furthermore, at high error rates and with competing TCP flows, a given TCP flow using ETEN performs better than the same flow not using ETEN. At high error rates, the TCP flows using ETEN are sometimes able to gain a much higher share of the link than they would otherwise, even when the competing non-ETEN flows traverse a shorter path.

However, the gains offered by per-packet ETEN schemes are insignificant in the presence of congestion. This is to be expected since the proposed ETEN schemes (correctly) defer to TCP congestion avoidance whenever congestion is detected. This means that ETEN is likely to offer the best benefits in error-prone networks that are uncongested (for example, rate-controlled or reservation-based networks).

Our work on the per-packet schemes provides useful baselines (upper bounds) for evaluating future ETEN proposals, both per-packet and cumulative. Under all circumstances, practical ETEN schemes should be required to perform no worse than stock TCP and they are not likely to perform better than Oracle ETEN.

We compared ETEN mechanisms to TCP Westwood which uses a bandwidth estimation strategy (instead of the AIMD strategy). We have observed that ETEN mechanisms slightly outperform TCP Westwood at very high error rates. At lower error rates, Westwood performs better even under congestion.

Per-packet schemes offer significant challenges to practical implementation by providing a new opportunity to exploit Internet security vulnerabilities and by requiring intermediate nodes to reliably extract information from the headers of corrupted packets. Cumulative ETEN appears to be more attractive to implement but may require protocol modifications.

We proposed, implemented, and evaluated a cumulative ETEN scheme that requires only a single scalar parameter to be tracked per link on each router. Our scheme requires additional packet header fields. Unfortunately, the particular scheme we evaluated did not realize the potential gains promised by per-packet schemes. We believe that further study is required to evaluate CETEN mechanisms.

Since accurate methods to infer the total loss for a TCP connection are not readily available, our CETEN strategy required tracking congestion losses at all intermediate routers, in addition to tracking corruption losses. We explicitly tracked congestion losses in our simulations, but the notification was conveyed differently from ECN [37]. Due to the limited success of our cumulative ETEN mechanisms in isolation, we did not study the interactions of ETEN with ECN.

We provide a discussion of the issues in dynamically adjusting FEC strength based on cumulative path error statistics. Masking of all losses by erasure codes (and the resulting need to separately track sequence numbers for the original data stream and the FEC-encoded stream) is a major practical challenge.

The primary challenges to ETEN deployment include need for modification to existing standards (which require

corrupted packets to be discarded early), security vulnerabilities, and the need for cross-layer co-operation. Security vulnerabilities include not only denial-of-service attacks but also more subtle attacks with effects ranging from unfair bandwidth sharing to total congestion collapse of the network.

8.2 Presentations, reports, publications, and software

The final report of this study will be made publicly available as BBN Technologies Technical Report Number 8333. We plan to submit the results of this research to a conference or journal.

The simulation software developed for this study as well as the Technical Report are available to researchers via the World Wide Web at *http://www.ir.bbn.com/projects/pace/eten/*.

Apart from BBN internal reviews, two presentations were made during the course of this study:

- The interim results from this study was presented to NASA GRC at a program review meeting in July 2001. The focus on cumulative ETEN mechanisms was influenced by the feedback received at this meeting.
- The interim results were also presented at the End-to-End Research Group Meeting held at MIT in 2001. The group was interested in the results, though the group consensus was that it was not clear from the data collected so far that ETEN gave enough value to be worth the deployment effort. This presentation did not include our results with cumulative ETEN or the comparisons to TCP Westwood (that were obtained later).

8.3 Recommendations for future work

This results of this initial broad study are intriguing; they lead us to recommend further work focused on specific aspects of ETEN. On the one hand, our work demonstrates tremendous potential from ETEN if reliable information extraction from headers were possible and congestion can somehow be controlled. On the other hand, it uncovers a number of practical challenges coupled with achieving only limited success with cumulative ETEN.

The primary thrust that we recommend is to explore cumulative ETEN alternatives that do not rely on congestion feedback from intermediate routers (since this would implicitly demand global deployment and render the scheme less practical). We believe that the biggest challenge to realizing CETEN schemes is the inability of a TCP endpoint to accurately estimate the total loss at a fine resolution (of a few packets) and in a timely manner (within an RTT to enable quick recovery). Research is needed to develop this capability.

Given this capability, we recommend that our proposed cumulative ETEN scheme should be refined to make use of it and then re-evaluated. The interactions of ECN with the refined cumulative ETEN scheme also remain to be studied in this context.

Future study of CETEN could focus on the impact the exact value of α has on performance, alternative corruption loss estimators, and alternative mechanisms (such as traceback messages) for end systems to collect corruption statistics from intermediate routers.

Our current effort focused on quantifying throughput improvements achievable using ETEN and was therefore limited to long-lived TCP flows. Further work is needed to isolate the effects of loss during the slow start phase and quantify the benefits of ETEN for short-lived flows. We also recommend that the mechanisms be evaluated using real network topologies and traffic traces including other workloads, for example, HTTP transactions.

Under high error rates, TCP connection establishment can be delayed or can fail completely. We believe that increasing the connection establishment rate under high error rates could be a key benefit of ETEN. We recommend that future work addresses this issue.

In future work, we recommend that the fairness of cumulative ETEN to other TCP flows be studied in detail. Further insights into the workings of CETEN can be obtained by isolating the performance penalty on mistaking congestion as corruption. The Oracle ETEN construct can be modified to enable this study.

We recommend that the effectiveness of ETEN be evaluated under other error models such as bursts and channel fades. We also recommend that the interactions of ETEN with dynamic FEC be implemented and analyzed via simulations. A specific challenge that remains to be addressed is the need to separately track sequence numbers of the data stream and the encoded stream (to correctly decouple recovery from corruption losses while still being able to accurately signal congestion.)

Contingent upon successful development of an accurate total loss determination strategy for TCP, integration with cumulative ETEN, and satisfactory evaluation, it is important that a TCP/ETEN prototype including the required IP router support be implemented and tested for performance and analyzed for security vulnerabilities. An implementation (or at least a detailed design) will permit realistic evaluation of interoperability with (and reusability of) existing network protocols and security mechanisms. The construction of this prototype would necessitate a detailed requirements analysis of the cross-layer communications between the layer that detects corruption (typically the link and network layers) and the transport layer, which performs ETEN-based recovery.

References

[1] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, R.H. Katz, "A Comparison of Mechanisms for Improving TCP Performance Over Wireless Links ," *IEEE/ACM Transactions on Networking*, Volume 5, Issue 6, December 1997, pp. 756–769.

[2] H. Balakrishnan and R.H. Katz, "Explicit Loss Notification and Wireless Web Performance," *Proceedings of the IEEE Globecom Internet Mini-Conference*, Sydney, Australia, November 1998.

[3] S. Biaz, and N.H. Vaidya, "Distinguishing Congestion Losses from Wireless Transmission Losses: A Negative Result," *Proceedings of the Seventh International Conference on Computer Communications and Networks (IC3N)*, New Orleans, October 1998.

[4] ETEN ns-2 code, *http://www.ir.bbn.com/projects/pace/eten*

[5] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," *ACM Computer Communication Review*, Volume 26, Number 3, July 1996, pp. 5–21.

[6] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communication Review*, Volume 24, Number 5, October 1994, pp. 10–23.

[7] S. Floyd, and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Transactions on Networking*, August 1999, pp. 458–472.

[8] S. Gibson, "The Strange Tale of the Attacks Against GRC.COM," *http://grc.com/dos/grcdos.htm*.

[9] T. Henderson and R. Katz, "Satellite Transport Protocol (STP): An SSCOP-based Transport Protocol for Datagram Satellite Networks," *Proc. 2nd Workshop on Satellite-Based Information Systems (WOSBIS '97)*, Budapest, Hungary, October 1997.

[10] V. Jacobson, "Congestion Avoidance and Control," *Proceedings of ACM SIGCOMM '88*, Stanford, CA, USA, August 1988.

[11] M.A. Jolfaei, B. Heinrichs, and M.R. Nazeman, "Improved TCP Error Control for Heterogeneous WANs," *Proceedings of the IEEE National Telesystems Conference*, San Diego, CA, USA, May 1994.

[12] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang, "TCP Westwood: End-to-end Bandwidth Estimation for Efficient Transport over Wired and Wireless Networks," *Proceedings of MOBICOM 2001*, Rome, Italy, July 2001.

[13] A.J. McAuley, "Reliable Broadband Communication Using a Burst Erasure Code", *Proceedings of ACM SIGCOMM '90*, Philadelphia, PA, USA, September 1990.

[14] ns-2 simulator, *http://www.isi.edu/nsnam/ns/index.html*

[15] K.K. Ramakrishnan, and R. Jain, "A Binary Feedback Scheme for Congestion Avoidance," *ACM Transactions on Computer Systems*, Volume 8, Number 2, May 1990, pp. 158–181.

[16] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols", *ACM Computer Communication Review*, Volume 27, Number 2, April 1997, pp. 24–36.

[17] N. Samaraweera, and G. Fairhurst, "Explicit Loss Indication and Accurate RTO Estimation for TCP Error Recovery Using Satellite Links," *IEE Proceedings in Communications*, Volume 144, Number 1, February 1997, 47–53.

[18] N. Samaraweera, "Non-Congestion Packet Loss Detection for TCP Error Recovery Using Wireless Links," *IEE Proceedings in Communications*, Volume 146, Number 4, August 1999, pp. 222–230.

[19] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson, "TCP Congestion Control with a Misbehaving Receiver," *ACM Computer Communications Review*, Volume 29, Number 5, October 1999, pp. 71–78.

[20] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," *Proceedings of ACM SIGCOMM 2000*, Stockholm, Sweden, August 28 – September 1, 2000.

[21] N. Shacham and P. McKenny, "Packet Recovery in High-Speed Networks using Coding and Buffer Management", *Proceedings of IEEE INFOCOM '90*, San Francisco, CA, June 1990.

[22] A.C. Snoeren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, S.T. Kent, and W.T. Strayer, "Hash-Based IP Traceback," *Proceedings of ACM SIGCOMM 2001*, San Diego, CA, USA, August 2001.

[23] J. Stone, and C. Partridge, "When the CRC and TCP Checksum Disagree," *Proceedings of ACM SIGCOMM 2000*, Stockholm, Sweden, August 28 – September 1, 2000.

[24] TCP Westwood code, *http://www.cs.ucla.edu/ mvalla/tcpw/tcpw-ORIG-casetti/TCPW-Casetti.htm*

[25] ISI, "Internet Protocol," *Request for Comments: 791*, September 1981.

[26] ISI, "Internet Control Message Protocol," *Request for Comments: 792*, September 1981.

[27] J. Postel (editor), "Transmission Control Protocol," *Request for Comments: 793*, September 1981.

[28] F. Baker, ed., "Requirements for IP Version 4 Routers," *Request for Comments: 1812*, June 1995.

[29] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options," *Request for Comments: 2018*, October 1996.

[30] S. Kent, and R. Atkinson, "Security Architecture for the Internet Protocol," *Request for Comments: 2401*, November 1998.

[31] S. Kent, and R. Atkinson, "IP Encapsulating Security Payload (ESP)," *Request for Comments: 2406*, November 1998.

[32] K. Ramakrishnan, and S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP," *Request for Comments: 2481*, January 1999.

[33] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," *Request for Comments: 2581*, April 1999.

[34] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," *Request for Comments: 3031*, January 2001.

[35] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations," *Request for Comments: 3135*, June 2001.

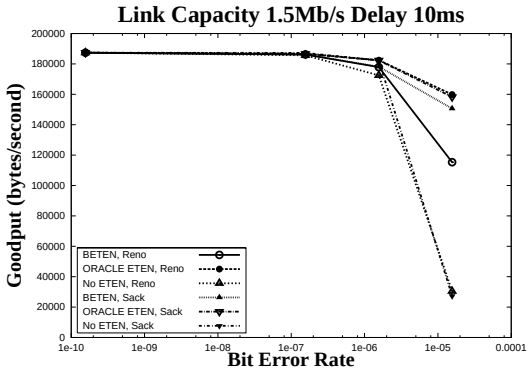
[36] S. Dawkins, G. Montenegro, M. Kojo, V. Magret, and N. Vaidya, "End-to-end Performance implications of Links with Errors," *Request for Comments: 3155*, August 2001.

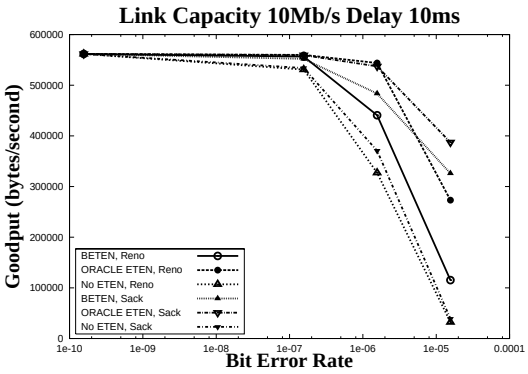
[37] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," *Request for Comments: 3168*, September 2001.

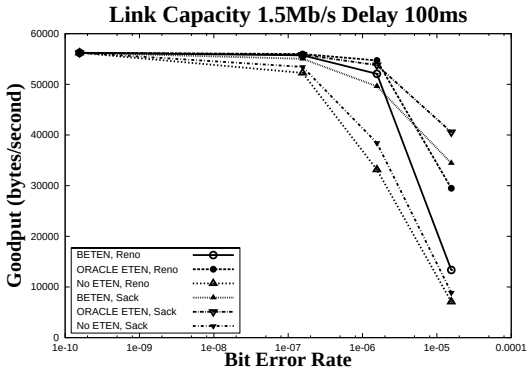
[38] B. Adoba, "IPsec-NAT Compatibility Requirements," *INTERNET-DRAFT, http://www.ietf.org/internet-drafts/draft-ietf-ipsec-nat-reqts-01.txt*, March 1, 2002.

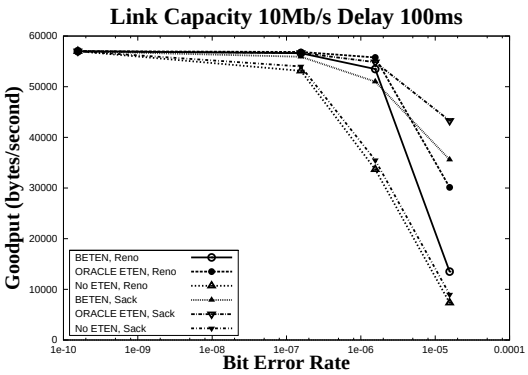
[39] S. Bellovin, M. Leech, and T. Taylor, "ICMP Traceback Messages," Internet-Draft *draft-ietf-itrace-01.txt*, October 2001.

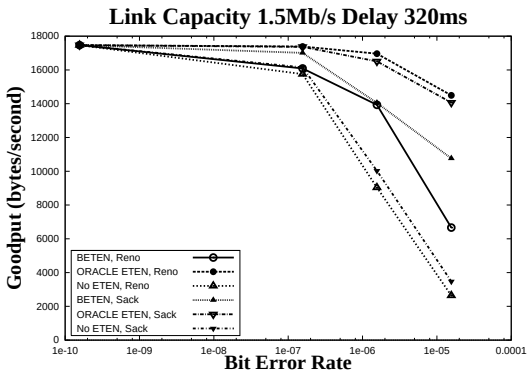
A Experiment 1: Single-hop topology with no cross-traffic

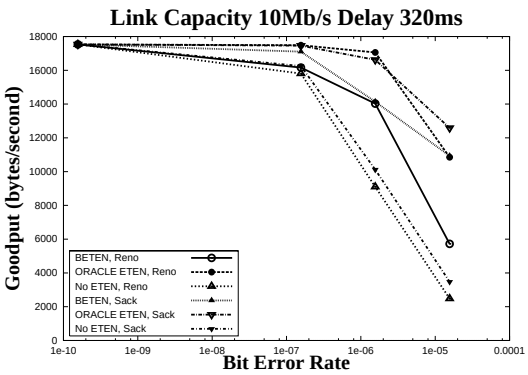


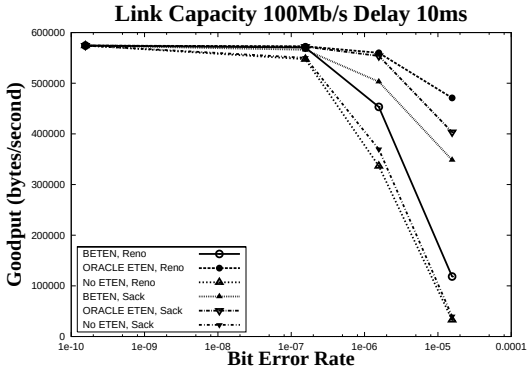


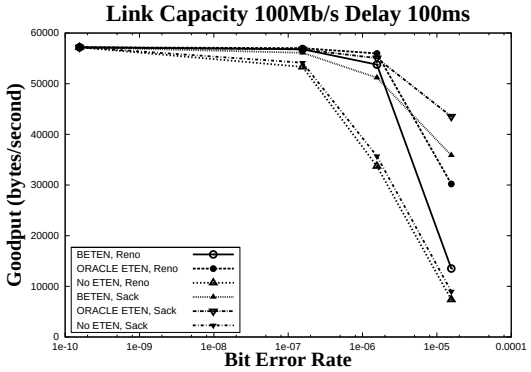


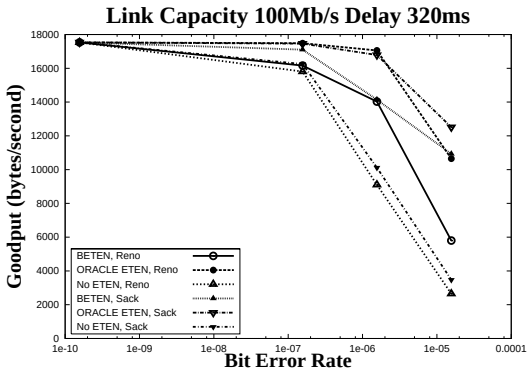




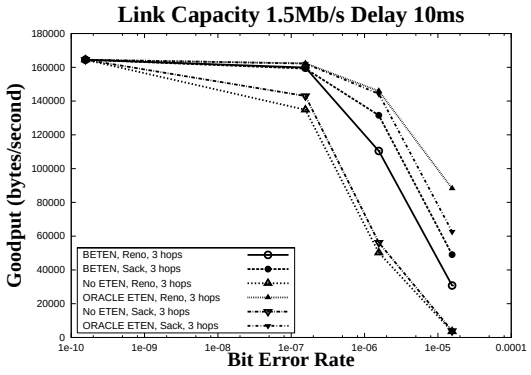


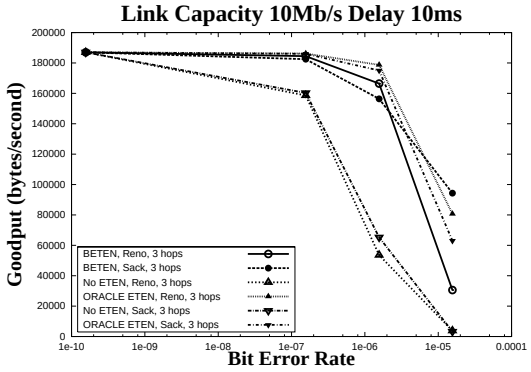

Link Capacity 100Mb/s Delay 10ms
Goodput (bytes/second)
Bit Error Rate
BETEN, Reno
ORACLE ETEN, Reno
No ETEN, Reno
BETEN, Sack
ORACLE ETEN, Sack
No ETEN, Sack

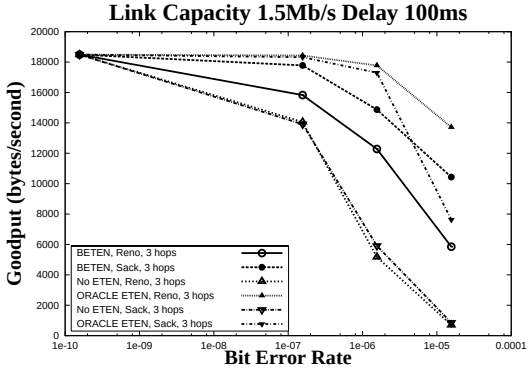

Link Capacity 100Mb/s Delay 100ms
Goodput (bytes/second)
Bit Error Rate
BETEN, Reno
ORACLE ETEN, Reno
No ETEN, Reno
BETEN, Sack
ORACLE ETEN, Sack
No ETEN, Sack

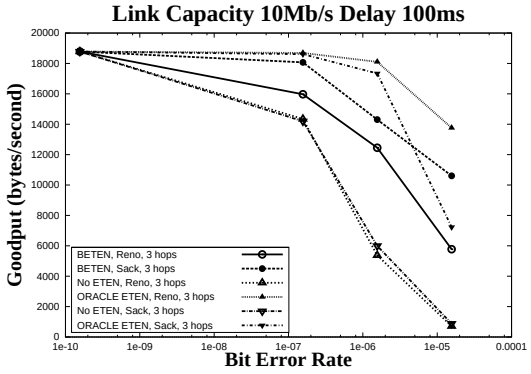

Link Capacity 100Mb/s Delay 320ms
Goodput (bytes/second)
Bit Error Rate
BETEN, Reno
ORACLE ETEN, Reno
No ETEN, Reno
BETEN, Sack
ORACLE ETEN, Sack
No ETEN, Sack

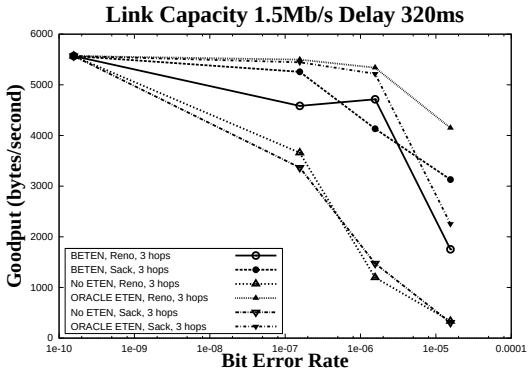
B Experiment 2: Multi-hop topology with no cross-traffic

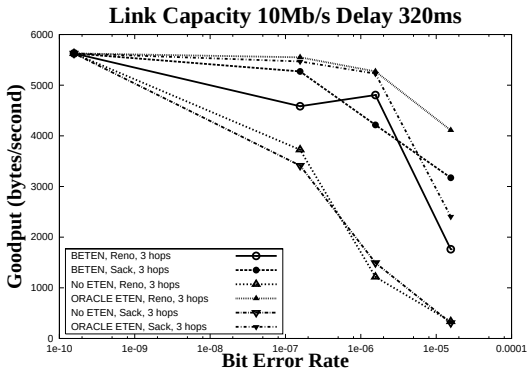


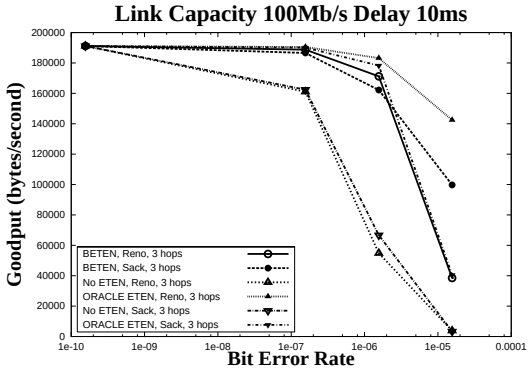


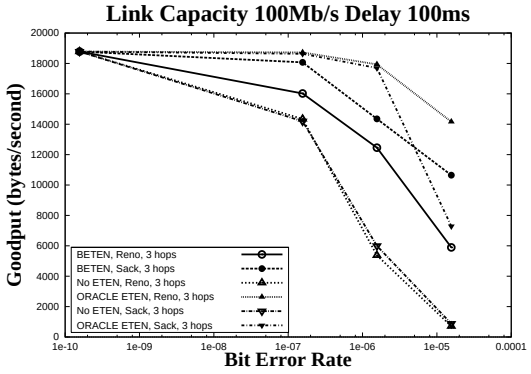


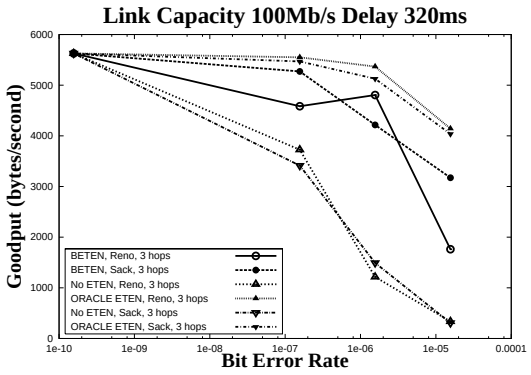




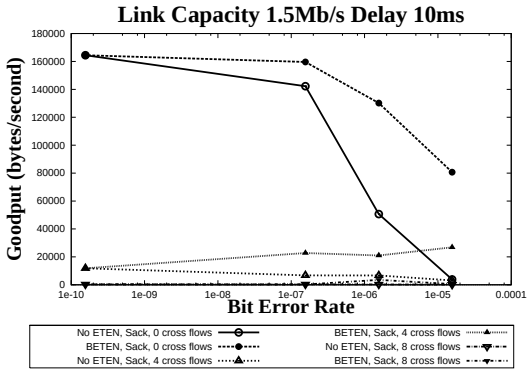


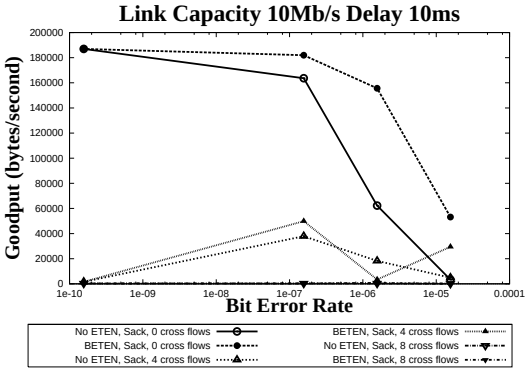

Link Capacity 100Mb/s Delay 10ms
Goodput (bytes/second)
Bit Error Rate
BETEN, Reno, 3 hops
BETEN, Sack, 3 hops
No ETEN, Reno, 3 hops
ORACLE ETEN, Reno, 3 hops
No ETEN, Sack, 3 hops
ORACLE ETEN, Sack, 3 hops

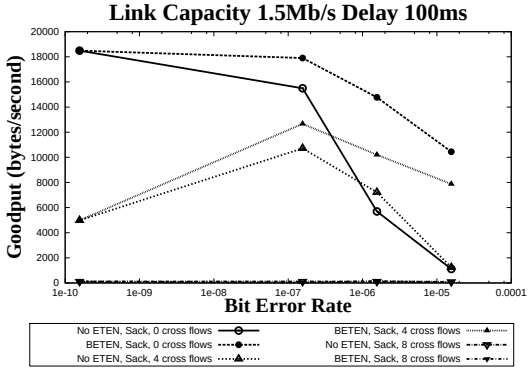

Link Capacity 100Mb/s Delay 100ms
Goodput (bytes/second)
Bit Error Rate
BETEN, Reno, 3 hops
BETEN, Sack, 3 hops
No ETEN, Reno, 3 hops
ORACLE ETEN, Reno, 3 hops
No ETEN, Sack, 3 hops
ORACLE ETEN, Sack, 3 hops

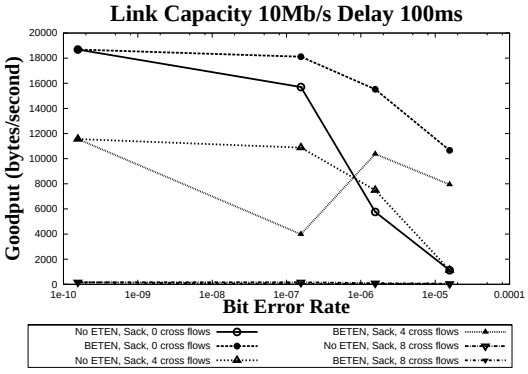

Link Capacity 100Mb/s Delay 320ms
Goodput (bytes/second)
Bit Error Rate
BETEN, Reno, 3 hops
BETEN, Sack, 3 hops
No ETEN, Reno, 3 hops
ORACLE ETEN, Reno, 3 hops
No ETEN, Sack, 3 hops
ORACLE ETEN, Sack, 3 hops

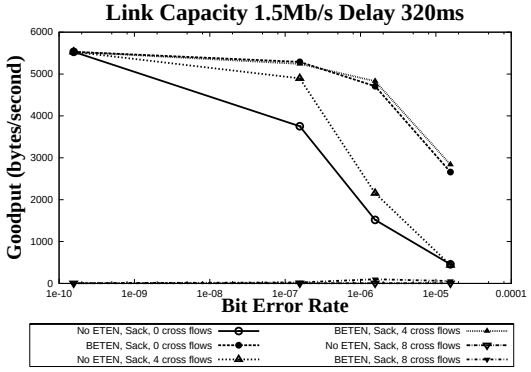
C Experiment 3: Multi-hop topology with UDP cross-traffic

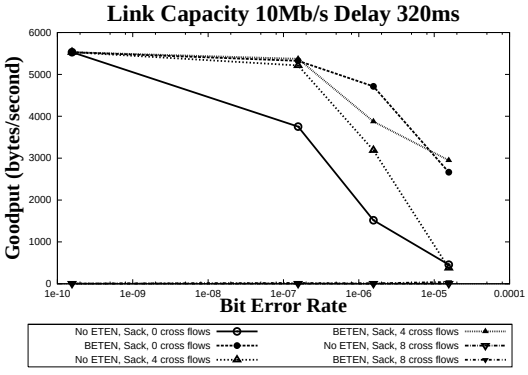


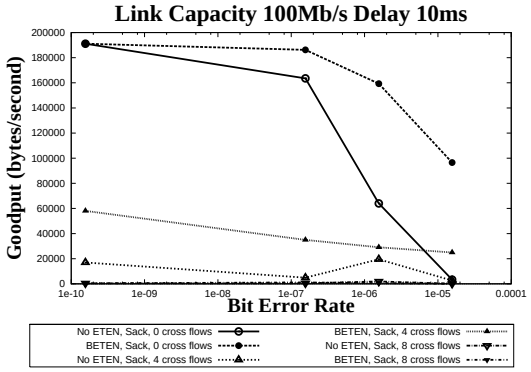


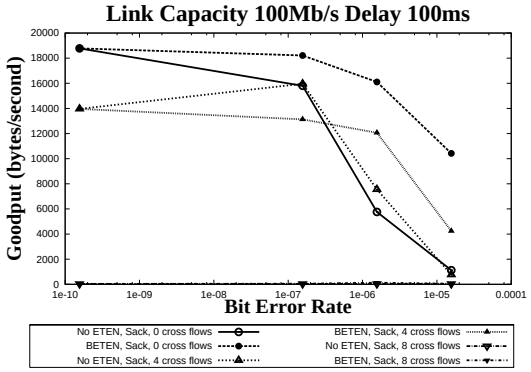


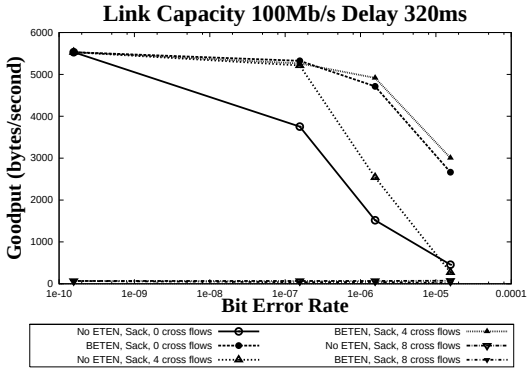




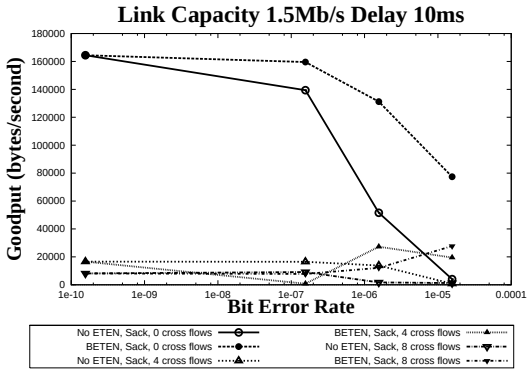


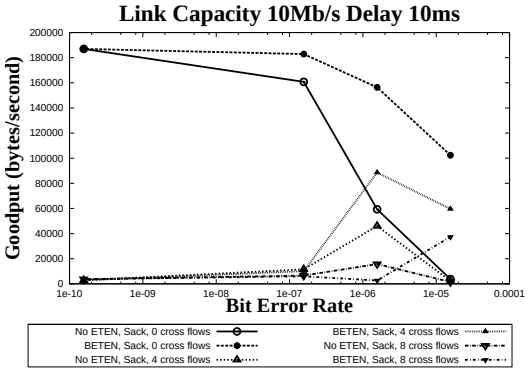

Link Capacity 100Mb/s Delay 10ms
Goodput (bytes/second)
Bit Error Rate
No ETEN, Sack, 0 cross flows
BETEN, Sack, 0 cross flows
No ETEN, Sack, 4 cross flows
BETEN, Sack, 4 cross flows
No ETEN, Sack, 8 cross flows
BETEN, Sack, 8 cross flows

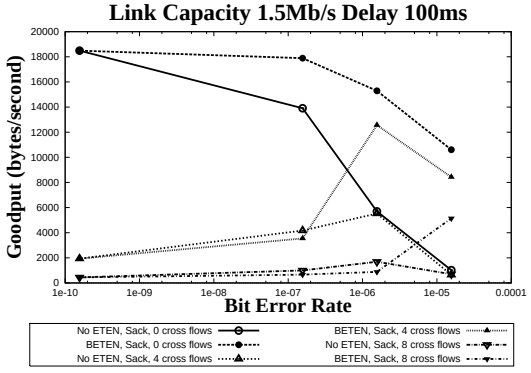

Link Capacity 100Mb/s Delay 100ms
Goodput (bytes/second)
Bit Error Rate
No ETEN, Sack, 0 cross flows
BETEN, Sack, 0 cross flows
No ETEN, Sack, 4 cross flows
BETEN, Sack, 4 cross flows
No ETEN, Sack, 8 cross flows
BETEN, Sack, 8 cross flows

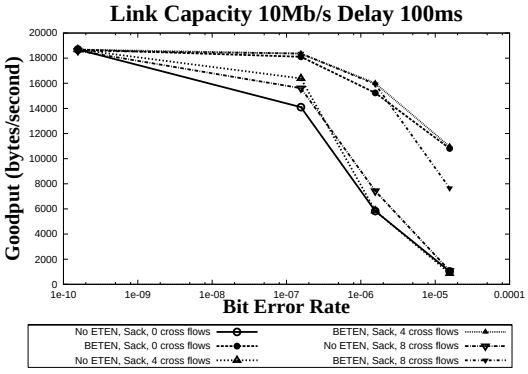

Link Capacity 100Mb/s Delay 320ms
Goodput (bytes/second)
Bit Error Rate
No ETEN, Sack, 0 cross flows
BETEN, Sack, 0 cross flows
No ETEN, Sack, 4 cross flows
BETEN, Sack, 4 cross flows
No ETEN, Sack, 8 cross flows
BETEN, Sack, 8 cross flows

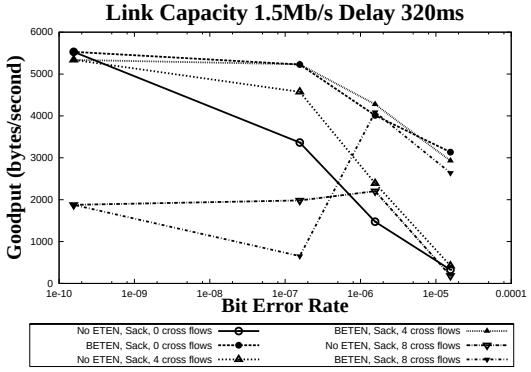
D Experiment 4: Multi-hop topology with TCP cross-traffic

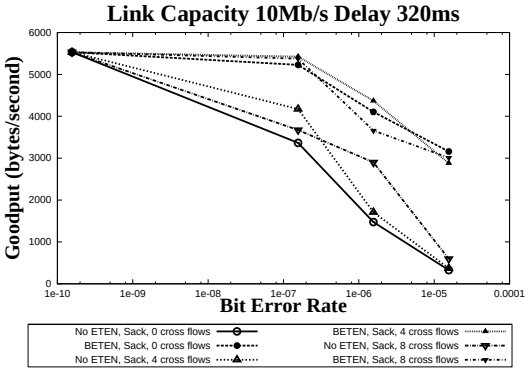


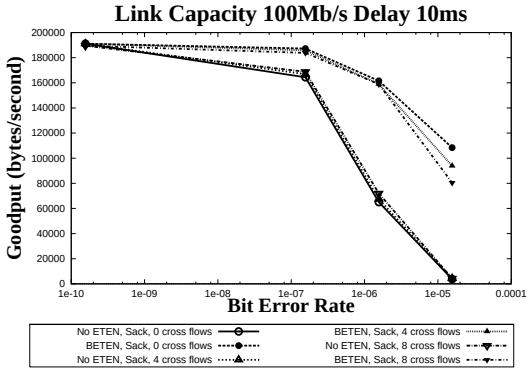


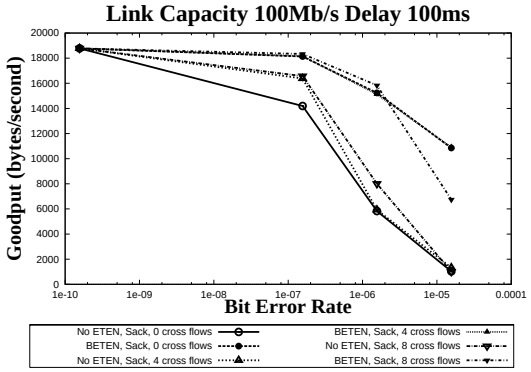


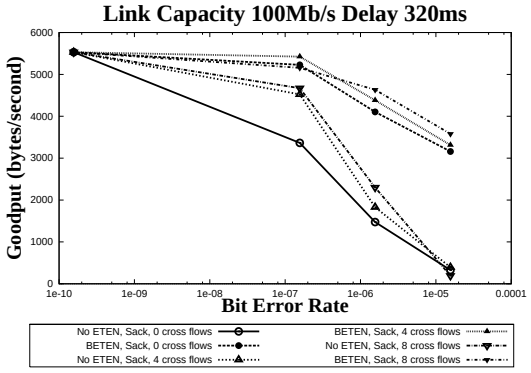




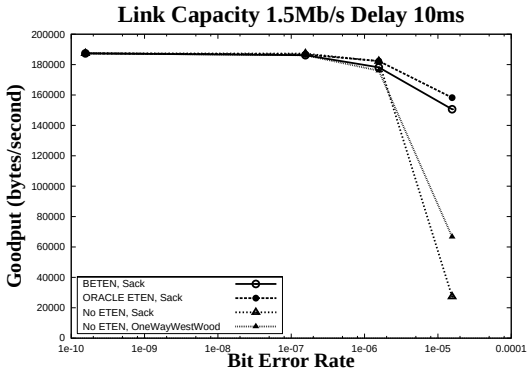


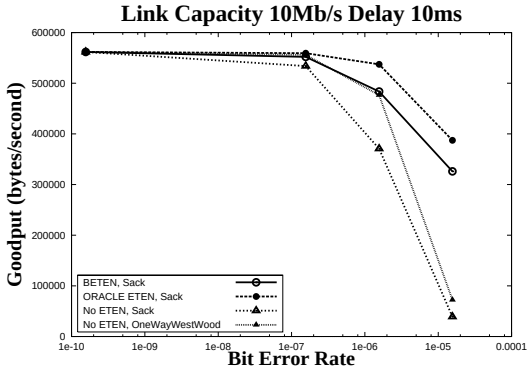

Link Capacity 100Mb/s Delay 10ms
Goodput (bytes/second)
Bit Error Rate
No ETEN, Sack, 0 cross flows
BETEN, Sack, 0 cross flows
No ETEN, Sack, 4 cross flows
BETEN, Sack, 4 cross flows
No ETEN, Sack, 8 cross flows
BETEN, Sack, 8 cross flows

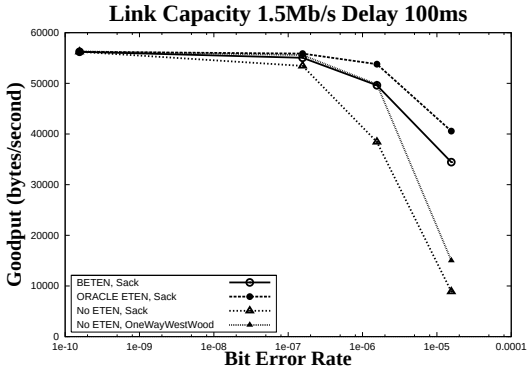

Link Capacity 100Mb/s Delay 100ms
Goodput (bytes/second)
Bit Error Rate
No ETEN, Sack, 0 cross flows
BETEN, Sack, 0 cross flows
No ETEN, Sack, 4 cross flows
BETEN, Sack, 4 cross flows
No ETEN, Sack, 8 cross flows
BETEN, Sack, 8 cross flows

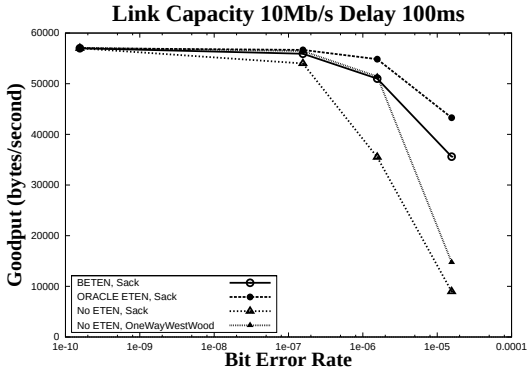

Link Capacity 100Mb/s Delay 320ms
Goodput (bytes/second)
Bit Error Rate
No ETEN, Sack, 0 cross flows
BETEN, Sack, 0 cross flows
No ETEN, Sack, 4 cross flows
BETEN, Sack, 4 cross flows
No ETEN, Sack, 8 cross flows
BETEN, Sack, 8 cross flows

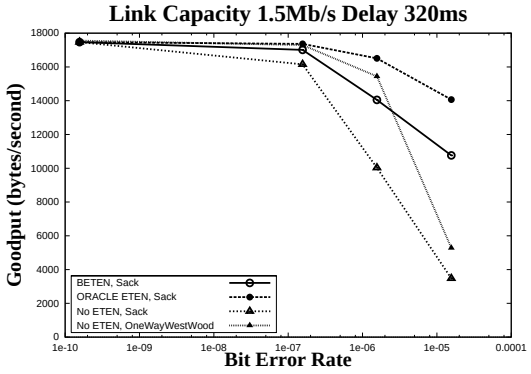
E Experiment 5: Per-packet ETEN versus TCP Westwood with no cross-traffic

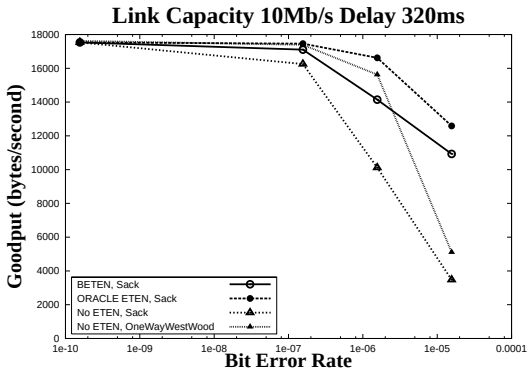


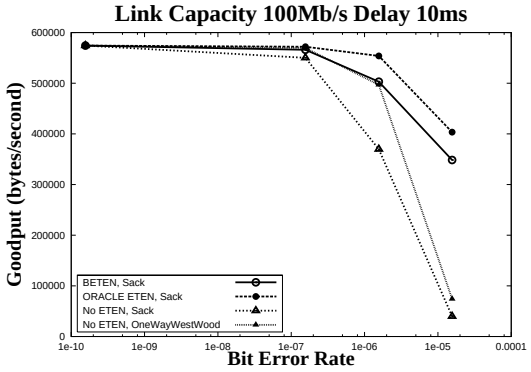


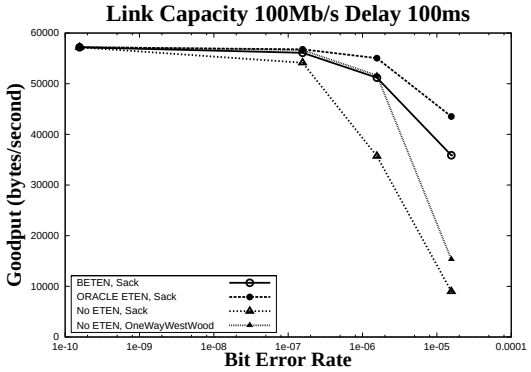


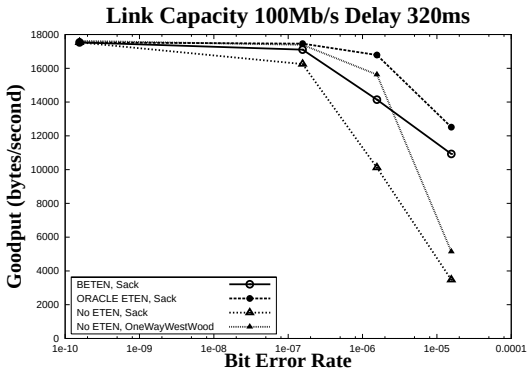




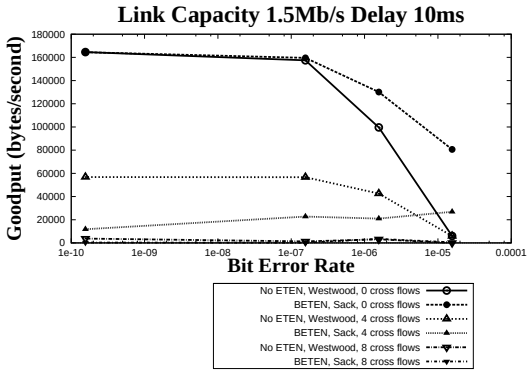


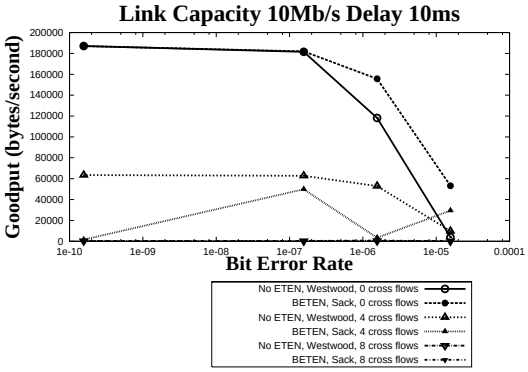

Link Capacity 100Mb/s Delay 10ms
Goodput (bytes/second)
Bit Error Rate
BETEN, Sack
ORACLE ETEN, Sack
No ETEN, Sack
No ETEN, OneWayWestWood

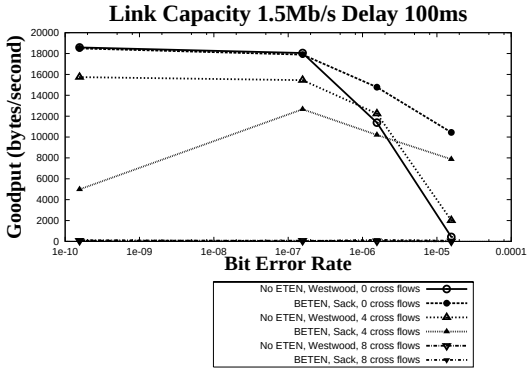

Link Capacity 100Mb/s Delay 100ms
Goodput (bytes/second)
Bit Error Rate
BETEN, Sack
ORACLE ETEN, Sack
No ETEN, Sack
No ETEN, OneWayWestWood

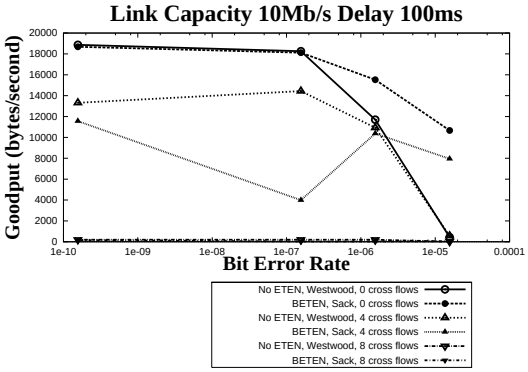

Link Capacity 100Mb/s Delay 320ms
Goodput (bytes/second)
Bit Error Rate
BETEN, Sack
ORACLE ETEN, Sack
No ETEN, Sack
No ETEN, OneWayWestWood

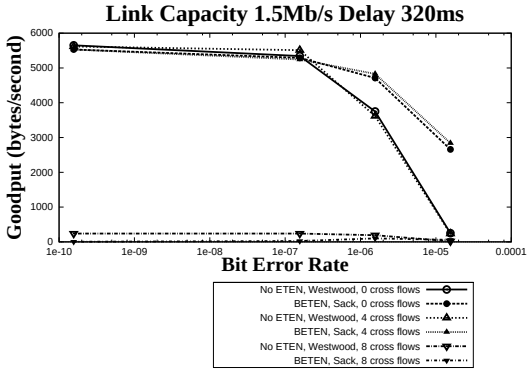
F Experiment 6: Per-packet ETEN versus TCP Westwood with UDP cross-traffic

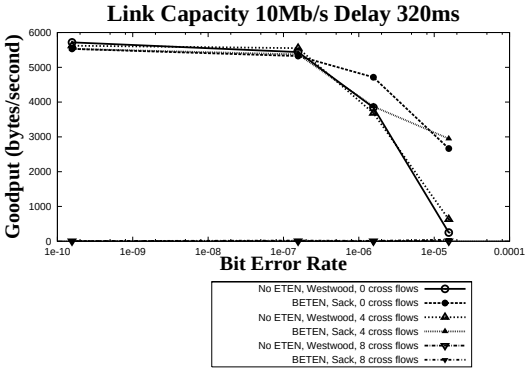


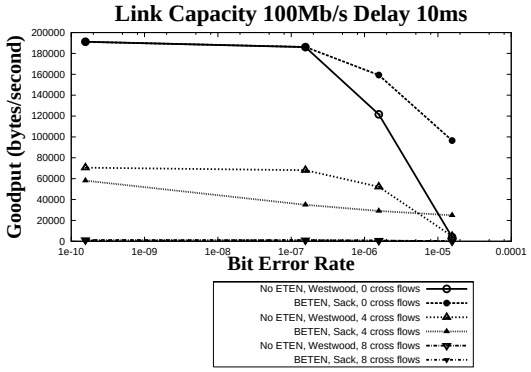


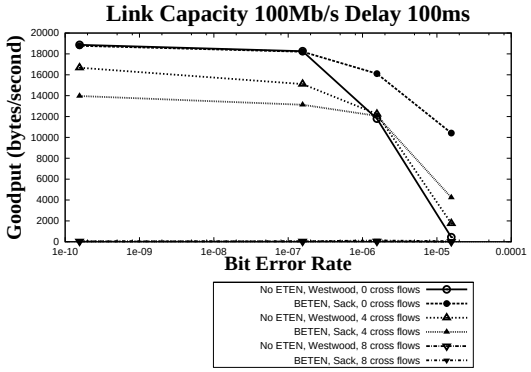


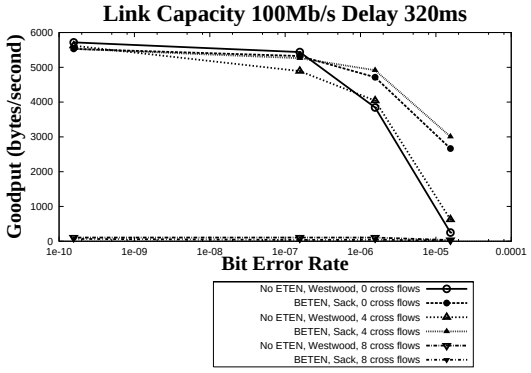




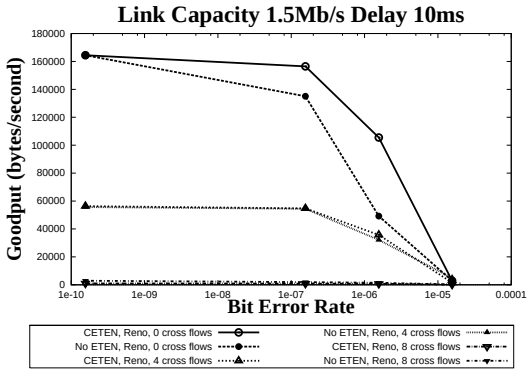


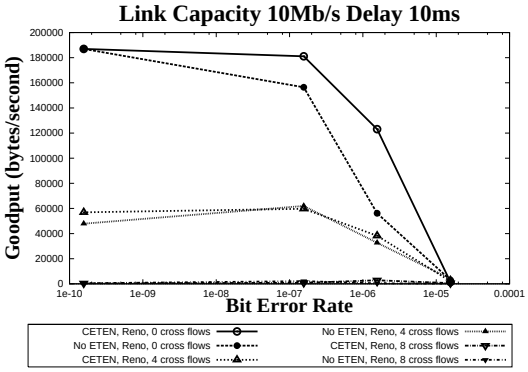

Link Capacity 100Mb/s Delay 10ms
Goodput (bytes/second)
Bit Error Rate
No ETEN, Westwood, 0 cross flows
BETEN, Sack, 0 cross flows
No ETEN, Westwood, 4 cross flows
BETEN, Sack, 4 cross flows
No ETEN, Westwood, 8 cross flows
BETEN, Sack, 8 cross flows

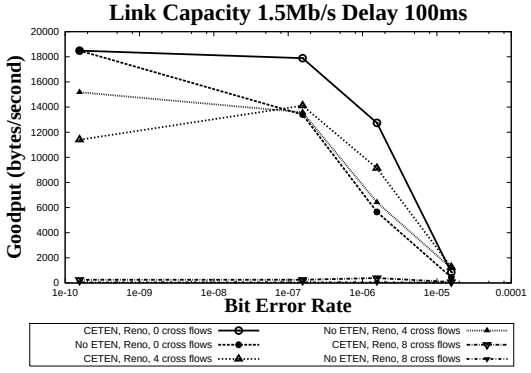

Link Capacity 100Mb/s Delay 100ms
Goodput (bytes/second)
Bit Error Rate
No ETEN, Westwood, 0 cross flows
BETEN, Sack, 0 cross flows
No ETEN, Westwood, 4 cross flows
BETEN, Sack, 4 cross flows
No ETEN, Westwood, 8 cross flows
BETEN, Sack, 8 cross flows

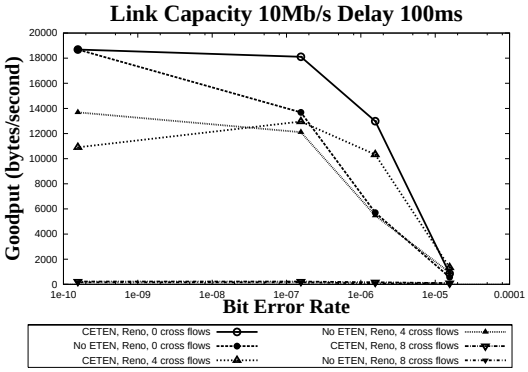

Link Capacity 100Mb/s Delay 320ms
Goodput (bytes/second)
Bit Error Rate
No ETEN, Westwood, 0 cross flows
BETEN, Sack, 0 cross flows
No ETEN, Westwood, 4 cross flows
BETEN, Sack, 4 cross flows
No ETEN, Westwood, 8 cross flows
BETEN, Sack, 8 cross flows

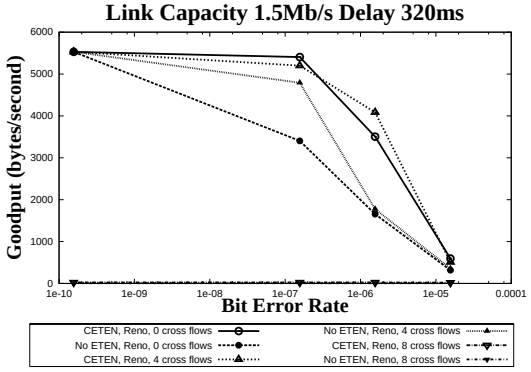
G Experiment 7: CETEN performance with UDP cross-traffic

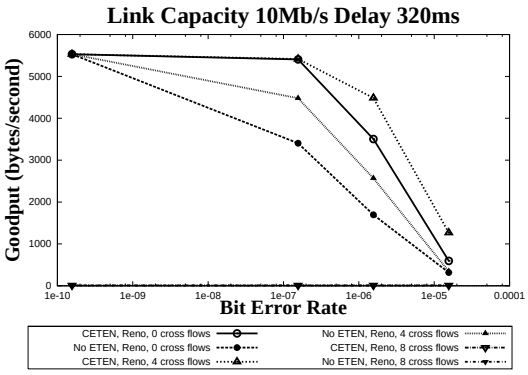


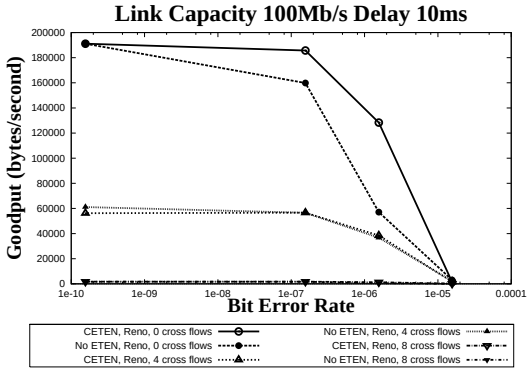


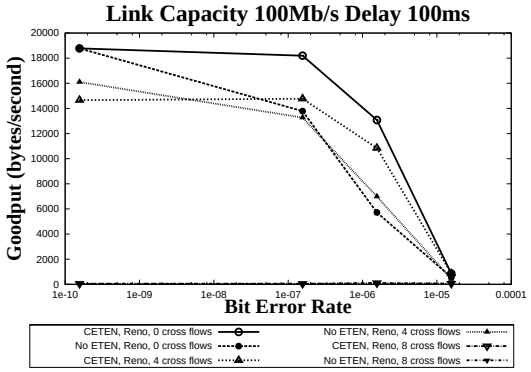


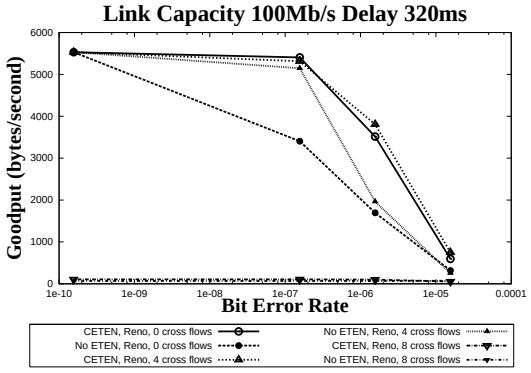




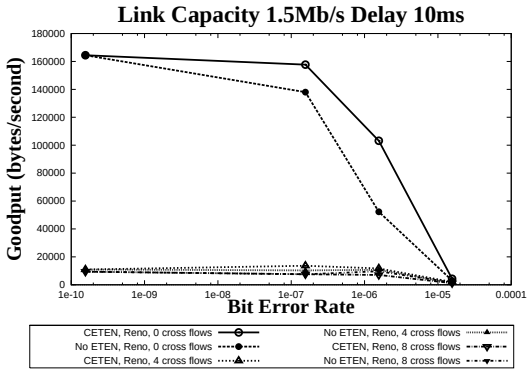


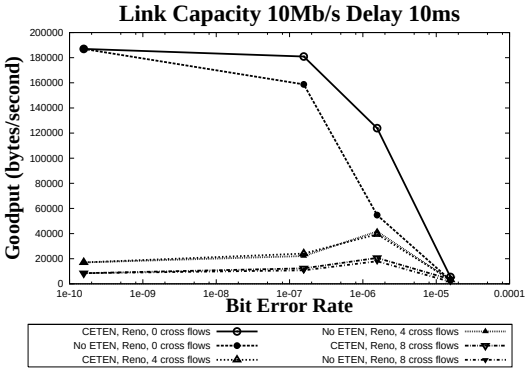

Link Capacity 100Mb/s Delay 10ms
Goodput (bytes/second)
Bit Error Rate
CETEN, Reno, 0 cross flows
No ETEN, Reno, 0 cross flows
CETEN, Reno, 4 cross flows
No ETEN, Reno, 4 cross flows
CETEN, Reno, 8 cross flows
No ETEN, Reno, 8 cross flows

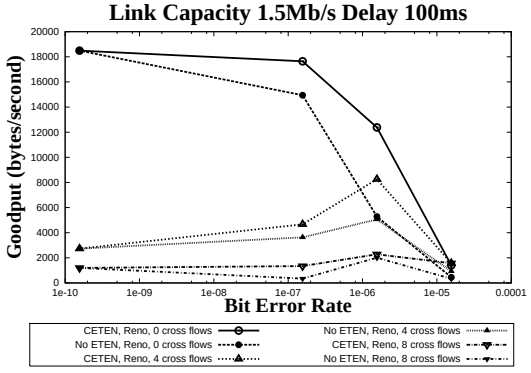

Link Capacity 100Mb/s Delay 100ms
Goodput (bytes/second)
Bit Error Rate
CETEN, Reno, 0 cross flows
No ETEN, Reno, 0 cross flows
CETEN, Reno, 4 cross flows
No ETEN, Reno, 4 cross flows
CETEN, Reno, 8 cross flows
No ETEN, Reno, 8 cross flows

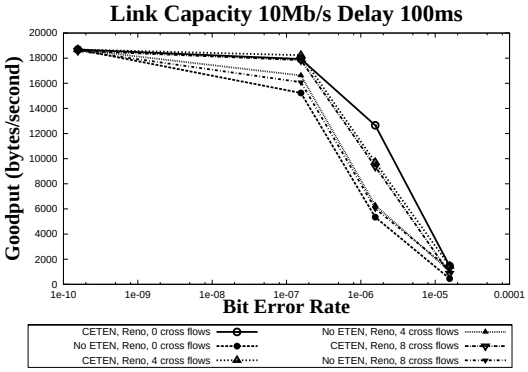

Link Capacity 100Mb/s Delay 320ms
Goodput (bytes/second)
Bit Error Rate
CETEN, Reno, 0 cross flows
No ETEN, Reno, 0 cross flows
CETEN, Reno, 4 cross flows
No ETEN, Reno, 4 cross flows
CETEN, Reno, 8 cross flows
No ETEN, Reno, 8 cross flows

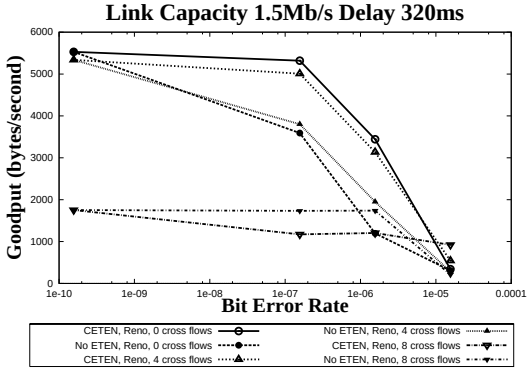
H Experiment 8: CETEN performance with TCP cross-traffic

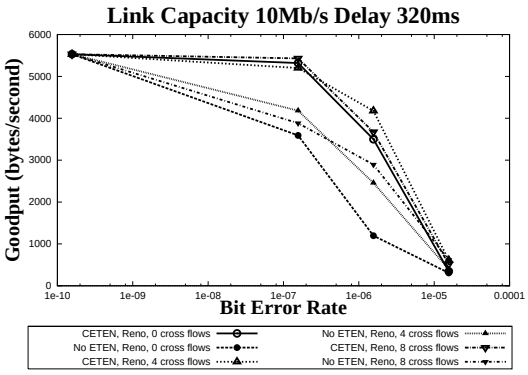


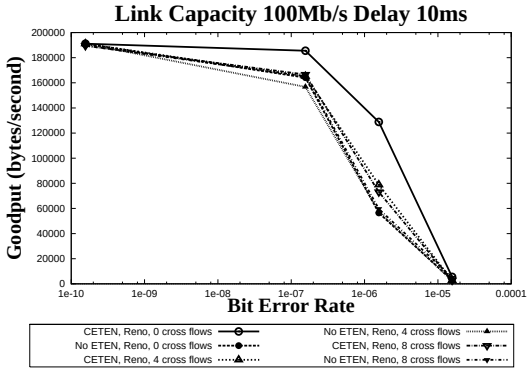


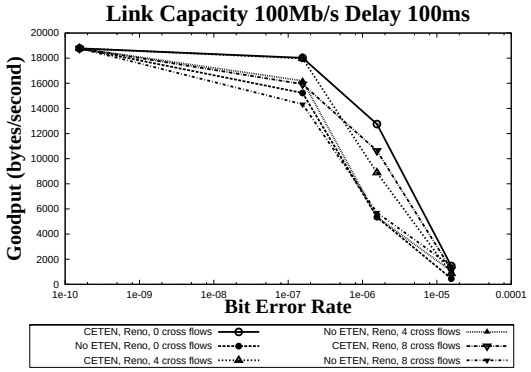


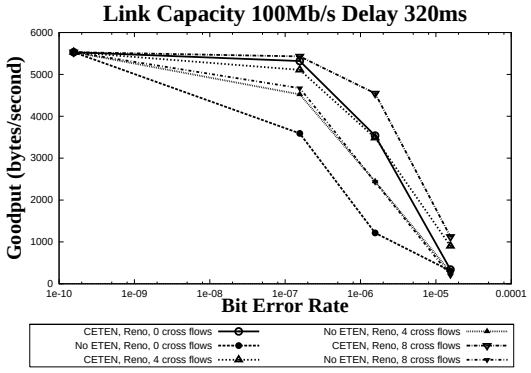







Link Capacity 100Mb/s Delay 10ms
Goodput (bytes/second)
Bit Error Rate
CETEN, Reno, 0 cross flows
No ETEN, Reno, 0 cross flows
CETEN, Reno, 4 cross flows
No ETEN, Reno, 4 cross flows
CETEN, Reno, 8 cross flows
No ETEN, Reno, 8 cross flows


Link Capacity 100Mb/s Delay 100ms
Goodput (bytes/second)
Bit Error Rate
CETEN, Reno, 0 cross flows
No ETEN, Reno, 0 cross flows
CETEN, Reno, 4 cross flows
No ETEN, Reno, 4 cross flows
CETEN, Reno, 8 cross flows
No ETEN, Reno, 8 cross flows


Link Capacity 100Mb/s Delay 320ms
Goodput (bytes/second)
Bit Error Rate
CETEN, Reno, 0 cross flows
No ETEN, Reno, 0 cross flows
CETEN, Reno, 4 cross flows
No ETEN, Reno, 4 cross flows
CETEN, Reno, 8 cross flows
No ETEN, Reno, 8 cross flows