An End-to-End Transport Protocol for Extreme Wireless Network Environments

Vijaynarayanan Subramanian, ECSE Dept, RPI, subrav@rpi.edu
Shivkumar Kalyanaraman, ECSE Dept, RPI, kalyas@rpi.edu
K.K. Ramakrishnan, AT&T Labs Research, kkrama@research.att.com

***Abstract*—**
As the Joint forces move towards the vision of network-centric warfare (NCW), it is extremely important that the network services be reliable and dependable, even under degraded network conditions. Tactical wireless and satellite based networks are prone to disruptions over multiple time-scales: bursty bit errors and packet loss (small time-scale), interference, jamming and capture effects (medium time-scale) and long-term path disruptions due to persistent channel impairments and mobility (large time-scale). TCP does not work well over such channels because it misinterprets erasure for congestion. TCP's throughput suffers significantly, particularly when there are disruptions. Large round-trip-times (RTT) as in satellite networks, and uncoordinated optimizations at multiple layers (PHY, MAC and transport) lead to poor performance.

In this paper we describe LT-TCP, an enhancement to TCP whick makes it robust and applicable for extreme wireless environments including a mix of ad-hoc meshed networks (MANETs), airborne networks and satellite networks. LT-TCP uses an adaptive, end-to-end hybrid ARQ/FEC reliability strategy and exploits ECN for incipient congestion detection. The novelty lies in our adaptive methods that respond to learning about the underlying random packet loss and disruption process. The overhead of FEC or smaller segments is imposed just-in-time and targeted to maximize the performance benefit (measured as improved goodput and timeout reduction) even when the path characteristics are uncertain. We show that LT-TCP substantially improves performance over regular TCP even for packet loss rates of up to 40% - 50%, thus substantially extending the dynamic performance range of TCP over lossy wireless networks.

I. Introduction

An example of the widespread use of wireless links in the current framework for information sharing in the DoD context is the communication among satellite relays, UAV/aircraft relays, ad-hoc microwave relays and ground stations/vehicles. Unlike organized commercial wireless networks, such networks are formed rapidly in battle conditions and therefore will face considerable uncertainty in terms of operating conditions (e.g.: RF environment, interference, jamming, capture effects, channel impairments). As a result of rapid and ad hoc deployment, the realized capacity compared to potential available capacity on such links may be unsatisfactorily low. Path disruptions could occur over multiple time-scales: small time-scales (bit or packet erasures), medium time-scales (interference, jamming, capture effect), long time-scales (persistent jamming, channel impairments, longer-term link failures). There are also situations where even when the path is functional without suffering a temporary disruption, end-to-end performance (capacity, delay, loss) could be highly volatile.

In such environments, performance variability is the norm: TCP will see variable capacity and unpredictable *residual* packet erasure rates (PER). Seamless communication under such conditions requires tolerance of such performance variability, especially packet erasures.

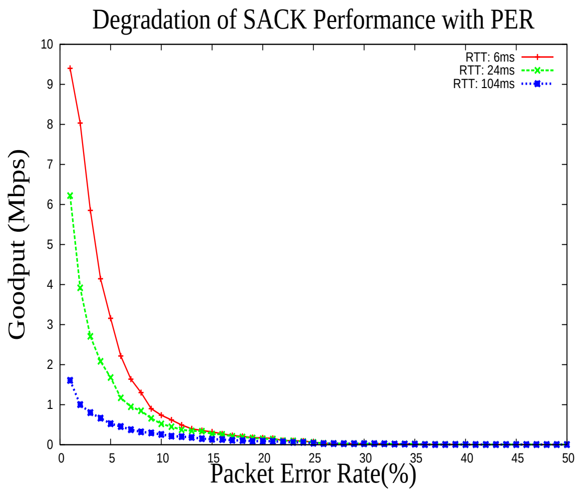


Fig. 1. TCP-SACK Degradation with Increased Erasure Rate and RTT (Uniform Loss Probabilities, 10 Mb/s Capacity, 1 flow)

TCP depends on packet loss to respond to congestion, and its drawbacks over lossy wireless links are well-known. A key issue is TCP's inability to distinguish between losses due to channel errors and congestion, leading to significant underestimation of the available capacity. This behavior only worsens as the channel error rate increases. It is important to separate TCP's response to congestion from packet erasures. As seen in Fig. 1, the performance of SACK deteriorates sharply as the error rate goes up. In practice, even an error rate of 5% is sufficient to cause a SACK connection to collapse. This behavior only worsens as the RTT increases.

Explicit Congestion Notification (ECN) is a mechanism that can be used to unambiguously indicate incipient congestion [8] and in conjunction with TCP's congestion avoidance mechanisms, it can reduce congestion loss (due to buffer overflow). ECN thus allows us to interpret and isolate packet losses as being primarily due to channel errors. The packet erasures due to errors on wireless channels still extract a substantial performance toll through TCP timeouts. In this paper, we re-examine

This work was funded by AT&T Labs Research, MIT Lincoln Labs/AFOSR Grant Letter No. 14-S-06-0206 and NSF-ITR 0313095.

TCP's behavior in ECN-enabled networks and propose adaptive mechanisms that allow robust performance even under heavy and persistent erasure conditions (e.g., up to 50% erasure rates). We propose a package of complementary mechanisms (adaptive MSS and proactive/reactive FEC) to recoup TCP's performance.

An interesting question is: Why end-to-end mechanisms for erasure tolerance over-and-above link-level error protection mechanisms? First, link level mechanisms may not be sufficient. Recently, studies by a group of researchers showed substantial *residual* performance variability (e.g., 10-50% packet erasure rates) in 802.11b mesh networks [5]. Emerging high speed LAN standards like 802.11n use adaptive modulation/coding techniques (i.e., variable capacity) targeting a packet error rate of 10%, but these techniques are triggered by low SNR events (i.e., bursty packet erasures). The efficacy of ARQ persistence in 802.11x is countered by exponential backoff timers, leading to substantial variability in capacity and delay.

Barakat et al. [1], [3] study TCP over links with just FEC or hybrid ARQ/FEC. They find a pure FEC strategy ineffective. Pure ARQ is also shown to fail for high erasure conditions, despite persistent retries. Though link-level hybrid ARQ/FEC is better than either FEC or ARQ alone, its performance also significantly degrades for higher loss rates (5% or more) despite high amounts of ARQ retries, fragmentation of IP packets, FEC overhead and buffering (see Fig. 15/16 in [3]). The situation is complicated further because different link layer standards/implementations have different erasure resilience capabilities.

Second, any appreciable residual erasures may have a disproportionate impact on TCP depending upon *which* packets are lost (e.g., data, acks, or retransmissions). Erasures of retransmissions or segments when TCP's window is small raise the risk of timeouts. In addition, information about the current window size, loss rate and packet size (MSS) can be exploited by TCP to provide the correct and variable amount of error protection when needed. Of course, our design (guided by the end-to-end design principle) does not preclude *general-purpose* error mitigation schemes at the link layer, and we remain cautiously optimistic about the potential of link-layer hybrid ARQ.

TCP Performance Enhancing Proxies (PEPs) [6] are TCP-aware mechanisms placed on boundaries where network characteristics change dramatically. PEPs maintain per-flow state and perform layer violations (with implications for security, mobility and scalability). The TCP-PEP technique is less applicable for the emerging regime of variable-performance, high-erasure, highly multiplexed, meshed wireless links.

Baldatoni et al. [9] proposed a version of TCP with FEC (but without adaptivity) that works with small error rates. Rizzo showed the feasibility of transport-layer high-speed FEC computation[14]. Although [14] mentions the idea of FEC in TCP, a specific scheme has not been studied and subsequent researchers' focus has been on multi-cast transport protocols [7], [11]. Bestavros et al. [4] propose a scheme called TCP Boston that aims to be tolerant to fragmentation in ATM networks by adding redundancy on a per-segment basis. Recent attempts at adding FEC to TCP have met with limited success [2] (for less than 10% erasure rates). Success with higher erasure rates have not been reported to the best of our knowledge. TCP Westwood [10] uses an estimate of output rate to guide congestion control, and has been effective for low erasure rates (under 5%). Presumably all these schemes encounter the risk of increased timeouts mentioned earlier. Overall, despite growing interest, there has been no clear baseline proposal that offers a significant increase in TCP performance over a wide range of erasure rates. Powerful and efficient error correction techniques have been proposed recently ([7]) that enable such operations to be done efficiently. In this work, we assume the use of Reed-Solomon codes [13] as the FEC mechanism.

In our scheme, called **Loss-Tolerant TCP (LT-TCP)**, we provision *proactive* FEC in the original window as a function of the estimate of the actual packet erasure rate. Subsequently, *reactive* FEC is used to mitigate the effect of erasures, during the retransmission phase. An adaptive maximum segment size (MSS) component provides a minimum granularity (a minimum number of packets) in the TCP window, again seeking to reduce the risk of timeouts. We seek to adaptively balance the FEC and packetization overhead while reducing the risk of timeouts and also rapidly recovering erased packets. In particular, when the end-to-end path has little or no loss/erasure, LT-TCP introduces minimal overhead. At the same time, we seek to significantly improve the performance of TCP and channel utilization even under packet erasure rates as high as 50% . An earlier version of LT-TCP and its performance evaluation appeared in [12]. We also evaluate the fairness of LT-TCP co-existing with TCP-SACK in this paper.

The rest of this paper is organized as follows. Section II describes the scheme. Performance results (ns-2 simulations) are presented in Section III. Section IV presents our conclusions and future work.

II. Scheme Description

LT-TCP design focuses on the following key issues:

- **Congestion Response:** How should TCP respond to congestion, but *not* respond to packet erasures. What is the appropriate signal of congestion in an error-prone environment?
- **Mix of Reliability Mechanisms:** What mix of TCP repair mechanisms (ARQ, FEC) should be used to achieve the TCP reliability objectives and how should they be structured?
- **Timeout Avoidance:** Timeouts are a final fall-back mechanism under significant congestion loss, but truly wasteful otherwise. How can the mix of TCP repair mechanisms be set up to reduce the timeout risk ?

Congestion Response: Our answer to this issue is simple: *react based only upon ECNs, not on detection of packet loss*. This solution would obviously work only in an ECN-enabled network. However, despite this simplifying assumption, timeout risk reduction poses further challenges as discussed below.

Reliability Mix: Error correction packets (a.k.a. FEC packets) have a property unlike regular data packets: if *any* k (out of N) packets are received, then it does not matter *which* k packets are received. A *unique* FEC packet can repair any one data

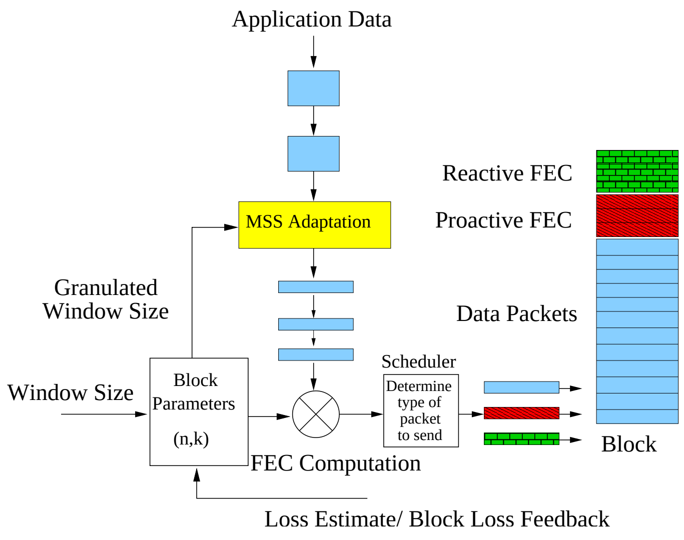


Fig. 2. LT-TCP Big Picture: PFEC and RFEC help in the recovery of data. PFEC operates in conjunction with Adaptive MSS and is determined by the current estimate of loss rate. RFEC is computed based on feedback from the receiver and the loss rate estimate. Incoming acks clock Data, PFEC and RFEC packets. The receiver can reconstruct the data packets as soon as any k out of n packets arrive at the receiver.

packet. In contrast, TCP uses SACK or 3-dupacks to identify and retransmit a packet with a *specific* sequence number. This sequence-agnostic property for FEC-based repair allows a unique FEC packet to be used either in the original window (i.e., in a proactive manner, called **PHASE 1**) or in the retransmission process (i.e., in a reactive manner, called **PHASE 2**). If the *cumulative number* of FEC and data packets in PHASE 1 and PHASE 2 do not meet the threshold of k (out of N), we will fallback to traditional retransmission or timeout. Our mix will first have adaptive amounts of proactive and reactive FEC repair packets, extending the traditional TCP mechanisms (SACK, dupacks, timeouts, retransmissions).

Timeout avoidance: Timeouts occur for the following key reasons that are exacerbated in a high packet erasure environment:
a) *All* packets in a window are lost.
b) *Three dupacks* do not reach the source (to trigger SACK-based repair).
c) One or more of the retransmitted packets are lost (because dupacks stop arriving).
To overcome each of these issues related to timeout avoidance, we propose to:
i) Granulate the TCP window more finely to increase the number of segments in a window that (due to the self-clocking nature of TCP) are spread over an RTT. Smaller packets also reduce the impact of bit errors (which translate to smaller packet error rates).
ii) Use proactive FEC packets in the window based upon an estimate of current erasure rate to reduce the need for dupacks and reduce the burden on SACK retransmissions for recovering lost packets.
iii) Use reactive FEC repair packets triggered by dupacks to complement and protect SACK retransmissions.

Overview: Fig. 2 provides an overview of the LT-TCP scheme. Application data is broken down into TCP segments where the MSS is chosen to accommodate proactive FEC (PFEC) packets in the window. Reactive FEC (RFEC) packets are computed at the same time and held in reserve. Feedback from the receiver provides not only the loss estimate but also information (e.g., SACK blocks) that can be used to compute the number of reactive FEC packets to send for each block. When the sender receives acks, it determines the type of packet to send (Data/PFEC/RFEC) and transmits them. This provides self-clocking and follows the semantics of TCP behavior. LT-TCP comprises the following building blocks that complement each other and extend SACK to provide resilience.

ECN-Only: Congestion response only to ECN, since it is a definitive signal of congestion in ECN-enabled networks.

Per-Window Erasure Rate Estimate (E): Since the amount of FEC overhead to include proactively depends upon the statistics of the loss process, we use an exponentially weighted moving average (A) of loss rate samples:

$$A = 0.5 * sample + 0.5 * A \tag{1}$$

The erasure rate estimation can be performed equally conveniently at either the receiver or the sender. The receiver can use the information from the packets received to estimate E while the sender can use information in the acks to do the same.

Proactive FEC: The number of FEC packets per window (P) used in PHASE 1 (i.e., Proactive FEC) is a function of the erasure estimate, i.e., $P = f(E)$. The MSS is adjusted to allow the desired number of FEC packets in the window (while maintaining sufficient window granulation). For example, if the estimated loss rate is 0.1, 10% of the packets in the following window are chosen to be PFEC packets. The remaining packets are data packets.

Adaptive MSS: Granulate the congestion window to have at least G (set to 10) packets, subject to limits of a minimum and maximum MSS (MSS_{min} and MSS_{max}). Depending on the window size in bytes, the MSS is adjusted to accommodate the required number of FEC packets while providing adequate erasure protection.

Thus, the adaptation of the MSS is governed by the following factors:

- The window must be divided into MSS-sized segments while maintaining the minimum granularity, G.
- The window should be able to accommodate at least $f(E)$ proactive FEC packets while providing adequate erasure protection for the estimated erasure rate, E.
- The MSS chosen must be bounded by the MSS_{min} and MSS_{max} values. Our current values are 200 and 1500 bytes respectively.

Reactive FEC: Reactive FEC packets are scheduled based on feedback present in incoming acks similar to data and proactive packets. Moreover, feedback from the receiver indicates the number of *holes* in each relevant block (alternatively, we could compute this using SACK blocks). This indicates the number of packets still needed at the receiver to decode packets from that block. The number of reactive packets scheduled is determined by the current loss estimate, number of holes the sender knows

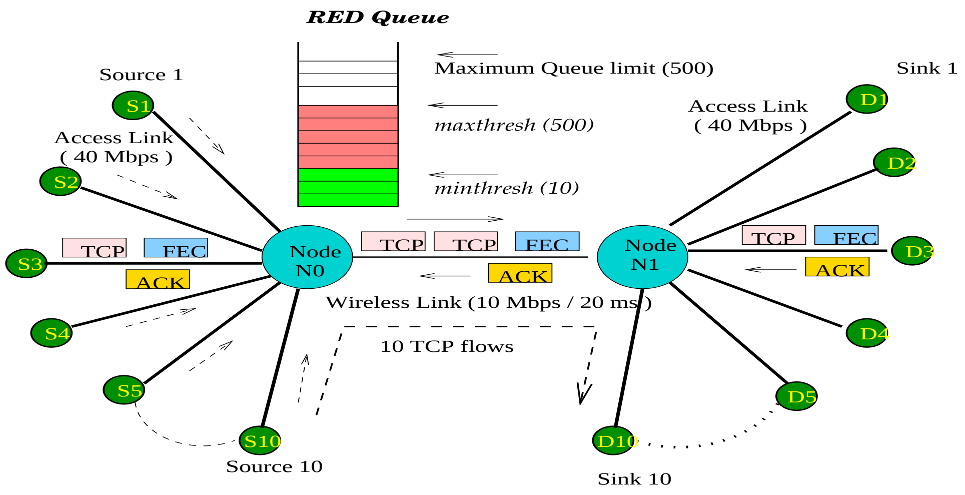


Fig. 3. Single Wireless Bottleneck Setup: RED AQM with ECN. 10 Mb/s bottleneck, 20 millisec. one-way delay, 10 TCP flows

about and the number of PFEC packets already sent. Reactive FEC packets thus scheduled are sent out in response to incoming acks. The specific type of packet sent (data /PFEC/RFEC) is determined by a *transmission scheduler* (see Fig 2). Moreover, following a timeout, sending RFEC packets before data allows the receiver to recover the block faster and protects data packets more efficiently. The reactive FEC packets complement and protect data in PHASE 1 and SACK retransmissions in PHASE 2.

The sender module is responsible for adaptive MSS adjustment (i.e., window granulation), computing proactive and reactive FEC packets, and the appropriate transmission of FEC packets.

The receiver implements packet reconstruction (using FEC if and when necessary) and per-window loss-rate estimation. The FEC overhead (proactive and reactive) is computed on a per-window basis using shortened Reed-Solomon (R-S) codes (similar to the method used in CD-ROMs). The proactive FEC is transmitted in the window, but the inventory of excess FEC packets is stored for potential use as reactive FEC.

The trade-offs of our mechanisms are as follows. Adaptive MSS uses smaller segments when windows are small and therefore the header (or packetization) overhead is larger, but diminishes as window sizes grow. Proactive FEC may lead to a small dead-weight goodput degradation due to over-estimation of erasure rate. However, since these mechanisms are all adaptive (i.e., they become more active only during higher erasure rate conditions), we argue that the trade-offs are worth making as they achieve a significant improvement in performance, and enables a wider dynamic range of applicability of TCP. The transmission of all the packets (DATA/PFEC/RFEC) is governed by the principles of self-clocking and LT-TCP adheres to the semantics of TCP in this regard.

III. Performance Results

In this section, we present the performance of LT-TCP compared with TCP-SACK (with ECN) and the performance of individual LT-TCP components. We also study fairness issues among TCP-SACK and LT-TCP flows. We use a single-bottleneck test case (see Fig. 3) with 10 flows and erasure rates varying from 0% to 50%. Hosts are ECN-enabled, bottlenecks implement RED/ECN on a 250 KB buffer (i.e., upto 500 packet of size 500-bytes). $minthresh$ and $maxthresh$ values are as shown. The results are based upon an average of 6 simulations, with each run lasting for 100 seconds of data-transfer to minimize effects of transients. The 95% confidence intervals (CIs) are also shown for key metrics. To assess the contribution of LT-TCP components, we use a 30% PER test case. Metrics include aggregate throughput, goodput, number of timeouts and congestion window dynamics. We account for all packet header overheads.

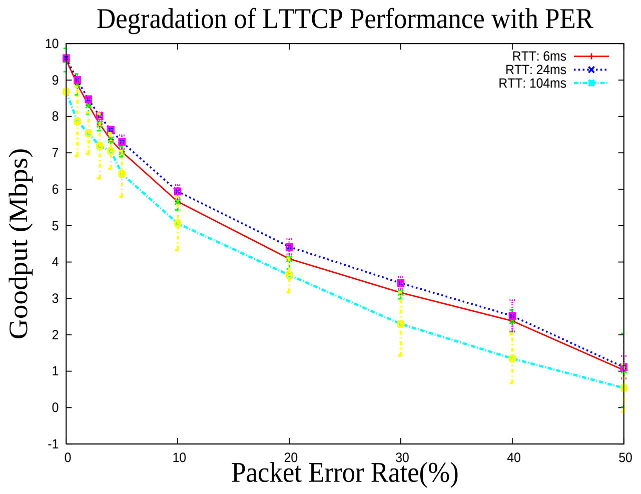


Fig. 4. LT-TCP performance with Increased Erasure Rate and RTT (Uniform Loss Probabilities, 10 Mb/s Capacity, 1 flow)

Error Models: We consider both a Uniform loss process and a two-state loss process with deterministic error periods (called Gilbert model) to test our scheme. We vary the average PER from 0-50% under both these loss models. We assume that the average erasure rate (p) is applied at the granularity of each packet. The ON and OFF periods have an erasure rate of $1.5 * p$ and $0.5 * p$. For p = 50% the ON-OFF period loss rates are 75%

and 25% respectively. This bursty model can be simplified to the uniform per-packet erasure model by setting the loss rates to be the average PER in both the ON and OFF states. In our simulations, the bursty loss model has ON/OFF periods with a mean of 10 ms, randomized over a small range (9-11ms).

A. LT-TCP vs TCP-SACK

Both TCP-SACK and LT-TCP perform well *without* packet erasures. But TCP-SACK's performance drops quickly for PER of 10% and higher. In practice, an error rate of around 5 % is sufficient to cause a single TCP-SACK connection to break down due to repeated timeouts. The number of timeouts experienced by SACK is much higher than with LT-TCP. With TCP-SACK, as the error rate goes beyond 40 %, while the number of timeouts decreases, the actual penalty is higher because repeated timeouts and timer back-off mechanisms cause timeout periods to increase exponentially. Fig. 4 shows the performance of a single LT-TCP flow at different error rates for a number of RTT scenarios. It can be seen that compared to the performance of TCP-SACK (Fig. 1), the degradation in performance is linear. LT-TCP manages to achieve better performance even at high packet error rates by avoiding timeouts and recovering lost packets using proactive and reactive FEC. Even when the RTT increases, LT-TCP achieves good performance, without the severe degradation observed with TCP-SACK.

Fig 5 shows the comparative performance with 10 flows for the Uniform and Gilbert loss processes. Multiplexing gains due to multiple flows enable SACK to obtain a goodput of around 3.02 Mb/s at 10 % PER (Uniform loss process) but performance beyond this error rate is dismal. In contrast, LT-TCP outperforms TCP-SACK by a wide margin and its absolute performance (goodput) is good up to about 50% PER. The sender-side throughput is close to the maximum achievable of 10 Mb/s. This improvement is in part due to the reduction in timeouts leading to smaller idle time. Moreover, the degradation in LT-TCP goodput is linear and it does not collapse as the error rate goes up. For example, at 10 % PER, we could potentially obtain a goodput of 9 Mb/s. LT-TCP manages to obtain around 6.49 Mb/s (for the Uniform loss process). The need to accommodate a minimum window granulation causes the header overhead in LT-TCP to increase compared to TCP-SACK. This overhead is due to multiplexing which reduces the available window size per LT-TCP flow. As the available bandwidth per flow increases, this overhead comes down. For example, with a single LT-TCP flow, the goodput is the same as that of SACK at 0% PER since LT-TCP can use full-sized segments while maintaining granulation. When the loss pattern is random and the number of PFEC/RFEC packets needed cannot be predicted perfectly, some amount of wastage of PFEC and RFEC occur.

B. LT-TCP Component Performance

The LT-TCP components are evaluated in the following (cumulative) order:

1) TCP-SACK.
2) TCP-SACK with ECN-only (i.e., RED/ECN at the bottleneck and congestion response only to ECN marks).

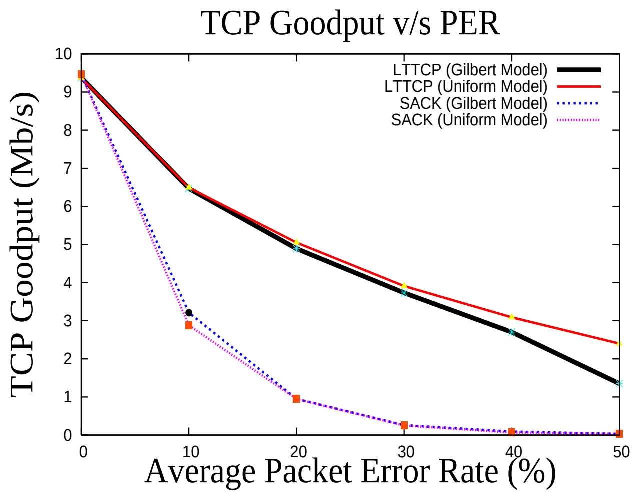


Fig. 5. LT-TCP and TCP-SACK performance with Increased Erasure Rates (Gilbert and Uniform Loss Probabilities, 10 Mb/s Capacity, 10 flows)

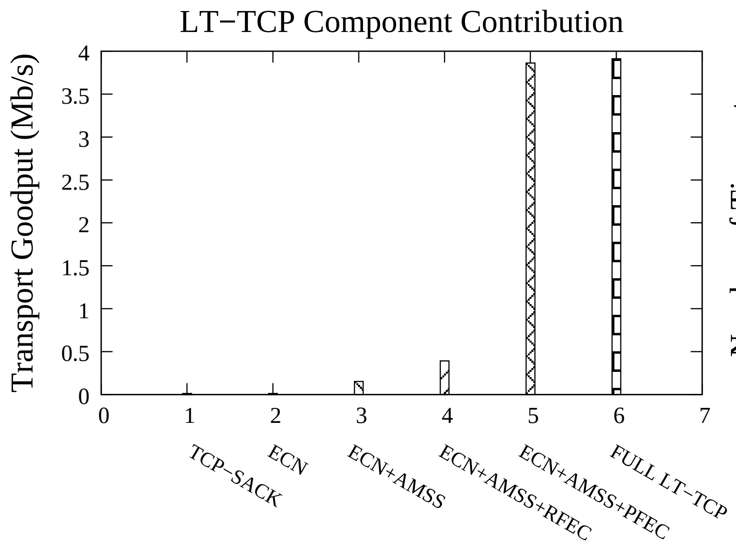
3) TCP-SACK with ECN-only and adaptive MSS.
4) TCP-SACK with ECN-only, adaptive MSS and reactive FEC (no proactive FEC).
5) TCP-SACK with ECN-only, adaptive MSS and proactive FEC (no reactive FEC).
6) Full LT-TCP scheme with TCP-SACK, ECN-only, adaptive MSS, proactive and reactive FEC.

The average goodput for the different component bundles is shown in Fig. 6(a) for a scenario with an error rate of 30% and an RTT of 50ms. The addition of each component to TCP-SACK consistently improves performance with the major gains being provided by the Proactive FEC protection. The final goodput for LT-TCP is several times the goodput achieved by TCP-SACK. The number of timeouts also decreases with the addition of each component. As mentioned earlier, while the number of timeouts with TCP-SACK is low, the length of each timeout increases exponentially, leading to low performance. With LT-TCP, timeouts are few and smaller in duration.

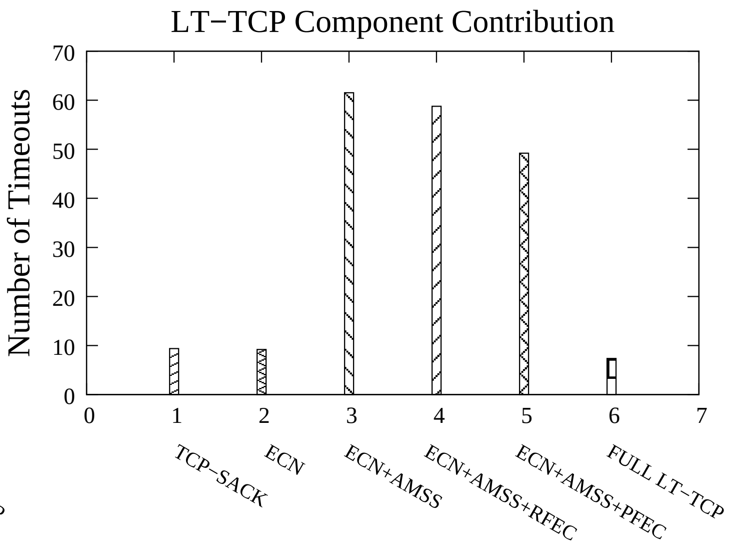
C. Fairness Among LT-TCP and TCP-SACK flows

We now evaluate the fairness of LT-TCP towards other TCP-SACK flows. Since TCP-SACK is unable to perform well (e.g., leaves the channel idle during timeouts) at even relatively small error rates ($> 5\%$), the available bandwidth in high loss scenarios may be utilized by LT-TCP. However, the comparison in the lossless scenario where the PER is 0% is also important. We test the fairness by sharing the bottleneck among 5 TCP-SACK and 5 LT-TCP flows. LT-TCP obtains an average goodput of 0.81 Mb/s while TCP-SACK obtains 1.10 Mb/s. Since the average packet size with LT-TCP is lower due to packetization overhead, LT-TCP's goodput is slightly lower. Overall, LT-TCP behaves fairly towards other TCP-SACK connections.

To determine the dynamic response of TCP-SACK when operating in conjunction with LT-TCP, we now look at the time it takes for TCP-SACK to recover from a timeout in this mixed scenario. On an otherwise lossless path, we experience a loss burst for a small 100 ms interval at time t=50 seconds, where the PER is 50%. This leads to a single timeout at TCP-SACK



(a) Goodput Comparison



(b) Timeout Comparison

Fig. 6. Figures 6(a) and 6(b) show the LT-TCP Component contributions at a packet error rate (PER) of 30 %.

just after this loss interval. The cumulative goodputs (measured as the goodput from the start of the simulations) in Fig 7(a) converge to give both flows an almost equal share of the bandwidth. The instantaneous goodput (measured for every 100ms window) in Fig 7(b) obtained by TCP-SACK dips sharply just after 50 seconds. However, TCP-SACK is able to recover from this timeout quickly. This is indicative of the fact that LT-TCP follows TCP semantics and TCP-SACK flows do not suffer due to LT-TCP. We see from Fig 7(c) that following the timeout, the congestion window is able to rise rapidly and reach its former level within a few RTTs. Although the LT-TCP connections experience losses, they do not suffer a timeout (see Fig 7(d)).

In summmary, LT-TCP's robustness does not lead to undesired aggressiveness and unfairness toward other flows. At high loss rates, where TCP-SACK is unable to perform, LT-TCP uses the available bandwidth. Under benign conditions, LT-TCP shares the bandwidth fairly with TCP-SACK connections.

IV. Summary and Conclusions

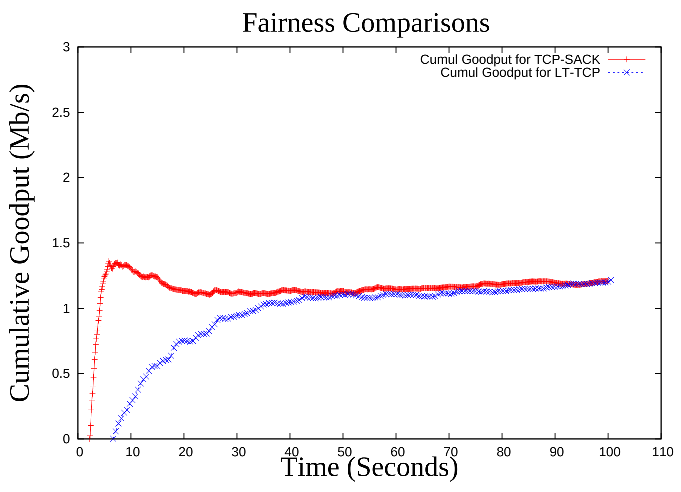
Transport protocols such as TCP have traditionally suffered poor performance in environments with lossy end-end paths. Wireless links in extreme environments such as military scenarios may experience jamming, interference and small/large time-scale outages leading to high end-end loss rates. To accommodate heterogeneity in links, multiple wireless hops and provide redundancy over longer time-scales, it is valuable to have a loss-tolerant transport layer that is not solely dependent on link layer mechanisms. Performance-enhancing proxies and other non-end-end solutions may be inapplicable in situations where security is a concern. At the same time, the transport protocol should not introduce overhead when it is unnecessary. These issues are currently relevant because of the growing use of meshed wireless networks and MANETs, *beyond their initial niches, as an integral part* of the future communications infrastructure.

Since TCP is the dominant reliable transport protocol used in the Internet, we have designed a loss-tolerant TCP (LT-TCP) which introduces additional mechanisms in an adaptive manner. Our enhancements allow good performance even under demanding conditions through the recovery of lost packets with Proactive and Reactive FEC packets that help avoid timeouts. In our performance evaluation, we demonstrated that LT-TCP improves the performance of TCP-SACK, for end-end packet error rates of even up to 50% while being fair to concurrent TCP-SACK flows . What is attractive about LT-TCP is that the achieved goodput shows a relatively smooth and linear decrease with increasing error rates, even with substantial end-end round trip times.

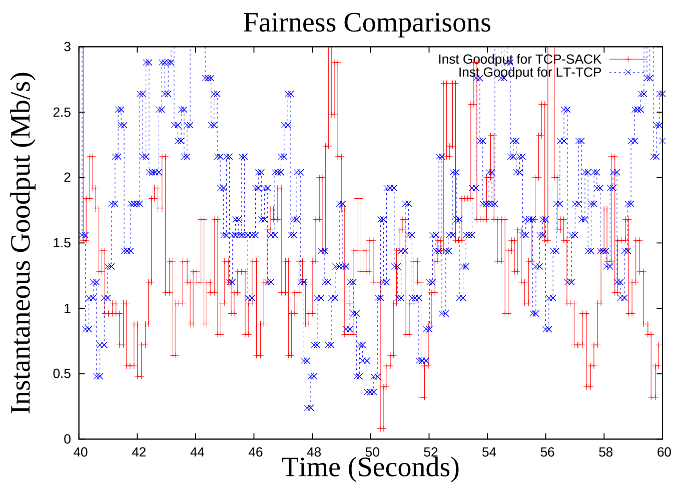
In our future work, we plan to complete our experimentation with LT-TCP in non-ECN environments, demonstrate backward compatibility as well as examine ways to make the protocol suite robust to longer time-scale outages while achieving reasonable goodput. We also propose to investigate link-level protocols that can help LT-TCP by performing local recovery and exporting a lower end-end loss rate. Such link-level protocols need to reduce the error rate while maintaining low latency (limited ARQ). Multi-hop paths with significant error rates on each link still pose a challenge and support from link-level protocols may be needed. We plan to investigate the division of mechanisms (between transport and link layers) to counter high loss paths in extreme networks and the performance of LT-TCP in conjunction with such link-level protocols, especially over multi-hop wireless networks. We also plan to explore the relative roles of link layer versus transport layer mechanisms, and where the balance, flexibility and cross-layer optimization opportunities exist for customizing our findings for specific DoD tactical network scenarios.

References

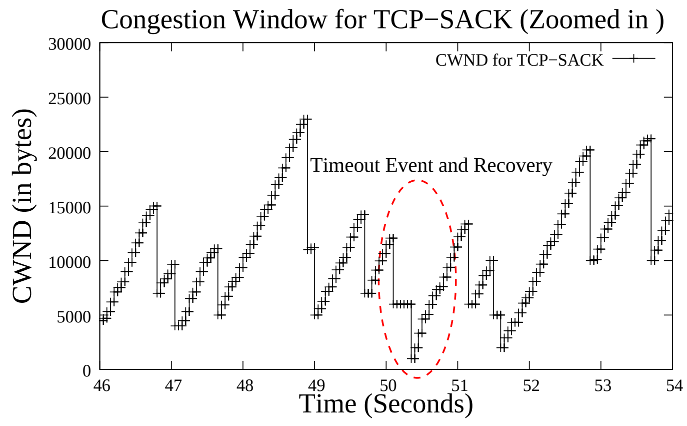
[1] C. Barakat and E. Altman. Bandwidth Tradeoff Between TCP and Link-



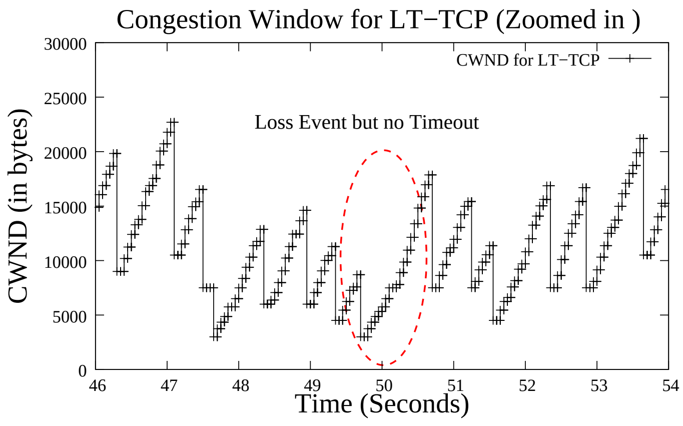
(a) Cumulative Goodput



(b) Instantaneous Goodput



(c) Congestion Window for TCP-SACK (Zoomed in)



(d) Congestion Window for LT-TCP (Zoomed in)

Fig. 7. The zoomed-in congestion windows (in bytes) for 1 TCP-SACK connection and 1 LT-TCP connection are also shown in the scenario where we have 5 connections of each type operating in a lossless environment. A 100ms loss period (PER of 50%) at 50 seconds causes a timeout in TCP-SACK. We plot the cumulative (measured from the start of the simulation) and instantaneous goodputs (measured in intervals of 100ms) obtained by the TCP-SACK and LT-TCP connections.

level FEC. *Computer Networks*, 39(2):133–150, June 2002.

[2] T. Anker, R. Cohen, and D. Dolev. Transport Layer End-to-End Error Correcting. Technical report, The School of Computer Science and Engineering , Hebrew University, 2004.

[3] C. Barakat and A. Fawal. Analysis of Link-level Hybrid FEC/ARQ-SR for Wireless Links and Long-lived TCP Traffi c. volume 57, pages 453–476, Amsterdam, The Netherlands, The Netherlands, 2004. Elsevier Science Publishers B. V.

[4] A. Bestavros and G. Kim. TCP Boston: A Fragmentation-Tolerant TCP Protocol for ATM Networks. In *INFOCOM '97: Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, page 1210, Washington, DC, USA, 1997. IEEE Computer Society.

[5] S. Biswas G. Judd D. Aguayo, J. Bicket and R. Morris. Link-level Measurements from an 802.11b Mesh Network. In *SIGCOMM*, August 2004.

[6] J. Griner G. Montenegro J. Border, M. Kojo and Z. Shelby. Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations. IETF RFC 3135, June 2001.

[7] M. Mitzenmacher J. W. Byers, M. Luby and A. Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. In *SIGCOMM*, pages 56–67, Aug-Sep 1998.

[8] K.K. Ramakrishnan, S. Floyd, D. Black. RFC 3168 - The Addition of Explicit Congestion Notifi cation (ECN) to IP, Sep 2001.

[9] L. Baldantoni and H. Lundqvist and G. Karlsson. Adaptive End-to-End FEC for Improving TCP Performance over Wireless Links. In *ICC*, June 2004.

[10] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links. In *Mobile Computing and Networking*, pages 287–297, 2001.

[11] J. Nonnenmacher and E. Biersack. Reliable Multicast: Where to Use FEC. In *Protocols for High-Speed Networks*, pages 134–148, 1996.

[12] O. Tickoo and V. Subramanian and S. Kalyanaraman and K.K.Ramakrishnan. LT-TCP: End-to-End Framework to Improve TCP Performance over Networks with Lossy Channels. In *Thirteenth International Workshop on Quality of Service (IWQoS 2005)*, Passau, Germany, June 2005.

[13] I. Reed and G. Solomon. Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8:300–304, June 1960.

[14] L. Rizzo. On the Feasibility of Software FEC. DEIT Technical Report LR-970131. Available as http://www.iet.unipi.it/ luigi/softfec.ps.