LT-TCP: End-to-End Framework to Improve TCP Performance over Networks with Lossy Channels

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

Abstract

As wireless channels are becoming common, the performance of TCP over networks with such links is important. TCP performance suffers substantially when packet error rates increase beyond a value of about 1% - 5%. This paper proposes an end-end mechanism to improve TCP performance over networks comprising lossy wireless link. The scheme separates the congestion indications from the wireless packet erasures by exploiting ECN. To overcome packet erasures we use a dynamic and adaptive Forward Error Correction (FEC) scheme that includes adaptation of the Maximum Segment Size for TCP. Redundancy is added in the form of proactive FEC which tunes itself to the measured error rate. The residual packet errors are handled by an enhanced retransmission scheme using reactive FEC repair packets to complement proactive FEC and SACK retransmission. Dynamically changing the MSS tailors the number of segments in the window for optimal performance. The scheme is built on top of TCP-SACK and depends on SACK and timeouts as a last resort. *ns-2* simulations show that our scheme substantially improves TCP performance even for packet loss rates up to 30%, thus extending the dynamic range and performance of TCP over lossy wireless networks.

I. Introduction

Data communication over wireless channels is becoming common. With the use of WiFi (802.11) hotspot/metro access, WiMax (802.16), 3G, mesh networks, community wireless networks and other wireless channels for data communication, end-to-end communication could involve traversal of multiple wireless links. As these emerging wireless channels grow in terms of nominal bandwidth capabilities, transport layers like TCP will see variable bandwidth, and unpredictable *residual* packet erasure rates. Seamless communication under such conditions requires end-end protocols to be tolerant of wireless link characteristics, including highly variable bit errors and packet erasures.

TCP, the predominant transport protocol in use today 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. The significant reduction of the congestion window in response to packet erasures causes TCP to severely underestimate the available bandwidth, thus underutilizing the available capacity. Indeed, TCP's capacity under-estimation worsens as the error rate on the channel increases. Therefore, when there is a bit error or packet erasure, it is desirable that TCP not react to it in the same manner as to congestion loss.

The use of Explicit Congestion Notification (ECN) to indicate incipient congestion allows us to isolate losses due to channel errors by sharply reducing congestion loss (due to buffer overflow). Our goal in this paper is to re-examine TCP's behavior in ECN-enabled networks and propose adaptive mechanisms that allow robust performance even under extremely heavy and persistent erasure conditions (e.g., up to 50% erasure rates). Thus, with TCP reacting to ECN [24], packet loss in a network with wireless links would be predominantly due to bit errors. However, the resultant packet erasures can still extract a substantial performance toll through TCP timeouts. We therefore

propose a package of complementary and adaptive mechanisms (adaptive MSS and proactive/reactive FEC) to reinstate TCP's performance. Our goal in devising these end-to-end mechanisms is to introduce as little additional protocol functionality inside the network, and to allow easy extension of TCP implementations. Our mechanisms allow robust TCP performance under extreme and highly variable erasure rate conditions. The available bandwidth utilization can be substantially improved.

An interesting question is: Why end-to-end mechanisms for erasure tolerance over-and-above link-level error protection mechanisms ? One motivation, which we also justify in the analysis presented here, is that the end-systems have the requisite information at the transport-layer, the ability to provision overhead and take corrective action *exactly* when packet erasures occur. For example, packets whose erasure can raise the risk of timeouts (e.g., retransmitted packets) are often known only to the end-systems, and may be protected with the introduction of FEC. In addition, information about the current window size, loss rate and packet size (MSS) are known at the end-system transport and can be exploited to provide the correct and variable amount of error protection when needed, as we show in this paper. These observations are consistent with the logic of the end-to-end design principle, because the best information needed at the right time for adaptive error protection is available at the end-systems. Of course, our design (or the end-end design principle) does not preclude *general-purpose* error mitigation schemes at the link layer, because our approach nicely complements such mechanisms to overcome any of their deficiencies. In particular, these mechanisms tend to provide a finite, and often fixed, amount of overhead (FEC or ARQ persistence) to recover from errors, so as to be suitable for a wide range of end-to-end transport protocols. Recent studies of link-layer FEC/ARQ [15], [14] have shown residual error rates that may be worse for vanilla TCP than for other transport protocols or applications (e.g., multimedia with their own coding schemes). In summary, while we acknowledge the role of link-level mechanisms to reduce average residual erasure rates, we focus on the emerging need to handle the end-to-end effects of such substantial residual erasure rates.

In this paper, we propose an approach that provisions *proactive* FEC on an end-end basis for TCP as a function of the actual packet erasure rate (PER) encountered. We also introduce a *reactive* FEC component to minimize the effect of erasures during the retransmission phase (in order to reduce the risk of TCP timeouts). The overall scheme, called **Loss Tolerant TCP (LT-TCP)**, includes an adaptive maximum segment size (MSS) component to provide a minimum granularity (a minimum number of packets) in the TCP window, once again seeking reduce the risk of timeouts. Our scheme seeks to adaptively balance the overhead added for FEC and the protection obtained for bit errors and the resultant packet erasures. When the PER estimate is low, the proactive and reactive FEC components adapt to reduce the amount of overhead added. Thus, when the end-to-end path has little or no loss/erasure, the FEC scheme introduces negligble 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 30-50 percent.

There has been substantial interest in the use of FEC in multicast transport protocols (e.g., digital fountain, Towsley et al, Nonnenmacher et al) and some TCP improvements over wireless networks ([35]). We fully survey these and other advances in the related work section. However, no prior work on TCP over wireless networks proposes a general, integrated, modular and adaptive solution that enables robust TCP performance in extreme and variable erasure environments (0-50% rates).

The rest of this paper is organized as follows. The next section examines related work and discusses the various proposals to improve TCP over wireless links, particularly the addition of FEC to TCP. Section III provides an overview of the proposed scheme and describes the sender and receiver mechanisms in greater detail. We present performance results from a detailed ns-2 simulation in Section IV. The last section presents our conclusions and

future work.

II. Related Work

There has always been anecdotal information about the high erasure and error rates in wireless links. Network designers have known that what matters to end-to-end protocols is the *residual packet erasure* characteristics of wireless links after any link-layer error mitigation is completed [25]. However, the situation with current standard link-level mechanisms is not encouraging. In a recent study, an MIT research group showed substantial variability in link performance in terms of capacity and erasure rates (e.g., 10-50% erasure rates) in 802.11b mesh networks [1]. Preliminary industry studies of WiMax and mobile/portable broadband access also suggest significant variability in performance [21]. Multi-hop ad-hoc networks used in defense or emergency response environments also exhibit such high residual erasure rates. These residual erasure rates (even after link layer error mitigation is done) have a substantial impact on end-to-end performance.

Let us consider the history of FEC and its use in transport/link layers. FEC has been well studied by communications technologists, especially as an error mitigation technique for digital voice communications [26]. Bit error correction is usually performed using convolution codes, turbo codes or a mix of coding and modulation [26]. In contrast, attractive building blocks for packet erasure correction include Reed-Solomon codes [27] and recently-proposed rateless codes [28].

FEC did not attract attention in the long evolution of TCP (1970 to mid-90s) primarily due to the computational costs of FEC and the fact that traditional retransmission/timeout mechanisms worked well over wired links. *Mildly* lossy links could be patched up with link-layer FEC at the bit or packet granularity. The problem really occurs when link level error mitigation fails due to substantial underlying error conditions. Luigi Rizzo in 1997 opened the field to software and kernel-level FEC by showing the feasibility of computing Reed-Solomon (R-S) erasure codes online at high speeds. Though Rizzo suggested possibilities for use of FEC in TCP, his and several subsequent researchers' focus has been on multicast transport protocols (for reasons unrelated to link-level errors). The digital fountain approach and rateless codes ([2]) improve on R-S codes for large block sizes in terms of computational complexity, and apply such codes to multicast transport. Nonnenmacher et al [6] developed analytical models of proactive and reactive FEC in the context of reliable multicast transport protocols (without considering congestion control issues). Rizzo and Huitema independently raised the possibility of using FEC for long delay and lossy wireless links (eg: satellite links), but do not propose a particular solution ([7], [8]). RFC 2488 [29] suggests link-level FEC for satellite links, leaving the end-to-end erasure-detection and response problem open.

FEC has also been considered with TCP recently, but with limited success. Anker et al [4] propose integrating (proactive) FEC with TCP, but since their method is neither adaptive nor does it consider MSS variation or reactive FEC, its performance gain is limited to lower erasure rates (less than 10%). Baldantoni [30] studies a simple FEC scheme in TCP based upon a slow moving estimator of current loss rates and reports marginal performance gains for up to 10% erasure rates. Performance gains for higher erasure rates have not been reported to the best of our knowledge.

Researchers have considered the implications of erasures on congestion control and counteracting mechanisms, again with limited success. In general, attempts to apply heuristics to distinguish between congestion and transmission error (e.g., using interarrival times and loss counts) have not been very successful ([18], [17]).

For LT-TCP we use ECN to signal congestion and an exponential moving average of erasure rates to adaptively tune the scheme parameters. In contrast, some researchers have proposed explicit loss/error/link quality notification

(ETEN [22], ELN [23], TCP-ELSA [9]). While we do not preclude the possibility of error notification techniques in the future, our choice of ECN allows us to operate end-to-end without introduction of any new functionality into the intermediate network nodes (ECN is already an IETF standard). Moreover, our work shows that there is more to the TCP/wireless challenge than distinguishing between loss and congestion: timeout risk reduction requires careful design of adaptive FEC and adaptive MSS building blocks.

TCP Westwood [19] measures output rate over a combination of low-to-medium bandwidth wired and modestly lossy wireless links and integrates this measure into TCP congestion control (decrease cwnd to this output rate estimate rather than an arbitrary halving). Again, this technique has been effective only for low erasure rates (under 5%), presumably due the increased timeout risks mentioned above.

TCP Eifel [13] is a scheme that deals end-to-end with issues such as sudden delay spikes in GPRS (2.5 G) networks using a modified RTO algorithm. This paper does not deal with heavy erasure conditions. We believe that a timeout risk reduction approach through FEC can be superior, and FEC also directly handles the recovery issues due to substantial erasures. In summary, the issue of TCP performance in an end-to-end manner over heavy erasure rate links remains open.

There is also considerable previous work on sophisticated link-layer techniques. One may put them in two broad categories: **a)** hybrid FEC/ARQ [15] and **b)** TCP performance-enhancing-proxies (PEPs) [11], [31], [32].

Hybrid FEC/ARQ involves fragmentation of packets (with its residual error-multiplying potential) and a *finite* degree of FEC/ARQ recovery at the link layer. Barakat and Altman [14] model TCP performance improvement with link-level FEC with block sizes of 10-40 packets and low erasure rates (less than 5%). They observe deadweight FEC overhead beyond a threshold and find that burst losses leads to residual erasures despite link-level FEC, and sharply reduced TCP throughput. Use of larger block sizes at the link-layer to improve FEC performance also poses challenges when TCP connections could be short or traffic unpredictable (an adequate backlog of packets may be unavailable).

Barakat and Al Fawal [15] study TCP over links employing hybrid ARQ/FEC. They report that ARQ mechanisms (like 802.11 ARQ with exponential backoff and limited persistence [33]) interfere with TCP timeouts and RTO estimation. 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 unreasonably high amounts of ARQ retries, fragmentation of IP packets, FEC overhead and buffering (see Fig. 15/16 in [15]). These studies and the MIT Rooftop measurements clearly indicate the limitations of link-level error resilience techniques.

Moreover, practice (in link-level error mitigation) diverges from potential. As wireless links have become standardized, faster and cheaper (e.g., 802.11), link level mechanisms have tended to be simple (e.g., 802.11's ARQ mechanism). This trend results in a high and variable residual erasure rate (e.g., 10-50% erasure rate observed by MIT's Roofnet project) that needs to be ultimately handled end-to-end. Different link layer technologies also have diverse capabilities for erasure resilience and this lack of consensus implies that the responsibility for overcoming residual erasures will ultimately rest with the transport or application layer. Furthermore, any appreciable residual erasure rate may have a disproportionate impact on TCP depending upon which packets are lost (e.g., whether they are data packets, acks, or retransmissions).

TCP Performance Enhancing Proxies (PEPs) [11] are TCP-aware mechanisms placed on boundaries where network characteristics change dramatically. PEPs include mechanisms for handling bandwidth asymmetry [31], [32], TCP-aware FEC provisioning [16] or adaptive link frame sizing [36]. PEPs, though effective in many cases,

maintain per-flow state and perform layer violations (with implications for security, mobility and scalability). The TCP-PEP technique is more applicable for last hop, highly managed, low-bandwidth, low-erasure links rather than the emerging regime of variable-performance, high-erasure, highly multiplexed, meshed wireless links that may appear not just as the last link of the path.

What about adaptive packetization mechanisms? The limitations of static packet sizes has also been mentioned in the literature. In a study of TCP over GSM data links (RLP), Ludwig and Joseph [34] point to the static MTU used end-to-end as a reason for diminished end-to-end error recovery performance over GSM links. Hybrid ARQ/FEC schemes internally choose unit sizes smaller than the advertized MTU [15]. The value of adapting the fragment unit sizes at the link-layer has also been observed recently [36]. Casetti et al [35] propose smaller packets for TCP during its initialization phase, but not subsequently.

Overall, there is growing interest in the use of FEC in transport protocols and improved link-layer techniques, but there is no clear baseline proposal (transport or link layer) that can offer a significant increase in TCP performance over a wide range of erasure rates (up to 50%).

III. Overview

LT-TCP design focuses on the following key issues:

- **Congestion Response:** How should TCP respond to congestion notifications, but not respond to packet erasures that do not signal congestion?
- **Mix of Reliability Mechanisms:** What mix of TCP repair mechanisms should be used to achieve the TCP reliability objectives ? In particular, what role would error correction (FEC) play in addition to traditional retransmission? How should the mix be split between proactive and reactive repair?
- **Timeout avoidance:** Timeouts, though useful as the final fallback mechanism, are truly wasteful in link utilization, application response times and increased risk of further timeouts. How should the mix of TCP repair mechanisms be setup to significantly reduce the probability of timeouts?

Our answer to the congestion response issue is simple: react only to ECNs. This solution would obviously work only in an ECN-enabled network. However, in spite of this simplifying network assumption, there are a number of residual timeout avoidance challenges as discussed below.

Error correction packets (referred to as FEC) have an interesting property unlike regular data packets: if k (out of N) packets are received to allow FEC reconstruction, then it does not matter *which* k packets are received. In other words, the dependence of repair by retransmission of a specific sequence number packet is effectively mitigated. A *unique* FEC packet can therefore protect or repair any one data packet. Contrast this with traditional retransmission: SACK or 3-dupacks will identify and require that a packet with a *specific* sequence number to be retransmitted. This sequence-agnostic property allows a unique FEC packet to be used either in the original window (i.e., in a proactive manner, **PHASE 1**) or in the retransmission process (i.e., in a reactive manner, **PHASE 2**). However, the cumulative number of sequence-agnostic FEC packets and sequence-specific data packets or retransmission packets in PHASE 1 and PHASE 2 have to meet the threshold of k for FEC to be effective. Else, TCP will revert to its fallback mechanism of traditional repair based on the sequence number of the retransmission or timeout, as discussed below. Our mix of reliability mechanisms will therefore include the traditional TCP mechanisms (SACK, dupacks, timeouts, retransmissions) combined with adaptive amounts of proactive and reactive FEC repair packets.

Timeout avoidance: we quickly review the reasons why timeouts occur and how these reasons are exacerbated in a high packet erasure environment:

- First, timeouts occur if *all* packets in a window are lost. We obviously do not solve this problem, but to reduce this risk, we propose to granulate the TCP window more finely. Given this finer granulation and TCP self-clocking, the packets in the window are spread more smoothly over an RTT (rather than in short bursts of packets).
- Second, even if only a subset of the window is lost, TCP timeouts will occur if three dupacks do not reach the source (the minimum required to trigger SACK-based retransmissions). If reverse paths also are subject to link erasures, then timeout probability increases further. With LT-TCP, we address this issue in two ways:

  a) by varying the maximum segment size (MSS) to allow the window to be more finely granulated (and hence have the capacity to generate dupacks), and,

  b) by provisioning 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 retransmission.
- Third, even if retransmissions are sent in response to SACK, the loss of ANY (one or more) of the retransmitted packets will cause TCP to timeout. We find that this retransmit-erasure sensitivity is a very important contributor to timeouts especially under high erasure rates. We address this issue by using reactive FEC repair packets triggered by dupacks to complement and protect SACK retransmissions. We trigger reactive FEC transmissions based upon dupacks (instead of SACKs) because we might face a situation where we do not get even three dupacks due to heavy erasures in the forward or reverse paths.

In summary, we propose four complementary building blocks (to extend TCP-SACK) in our scheme:

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

Adaptive MSS: Granulate the congestion window to have at least G packets, subject to limits of a minimum and maximum MSS.

Proactive FEC: FEC packets used in PHASE 1 are called Proactive FEC packets.

- Estimate per-window erasure rate E with a sensitive estimator. We use an exponential weighted moving average (EWMA), with adaptive parameters.
- Add P FEC packets per window as a function of the erasure estimate, i.e., P = f(E). The MSS is adjusted to allow one or more FEC packets per window (while maintaining sufficient window granulation).
- Compute FEC on a per-window basis using Reed-Solomon (R-S) codes. As a side effect, this method yields a large inventory of excess FEC packets for potential reactive FEC use (see below).

Reactive FEC: For every dupack, send R reactive FEC packets. R is a function of the erasure rate estimate, E. i.e., R = g(E). These reactive FEC packets will complement and protect SACK retransmissions in PHASE 2. Observe that there is nothing special about reactive FEC compared to proactive FEC: both of them are sequence-agnostic repair packets created as a function of estimated erasure rate, to protect data or retransmissions in PHASE 1 or PHASE 2 respectively.

The scheme details are captured next. We can further understand the complementary nature of the proposed building blocks by asking what would happen in their absence. Without the adaptive MSS mechanism, we would have insufficient window granulation, potentially resulting in insufficient dupacks to trigger retransmissions, larger packet erasure rates (PER) for a given bit-error-rate (BER), and difficulties in provisioning proactive FEC packets within the window for a targeted degree of overhead. Without Proactive FEC, there would be an increased reliance

Term	Detail	Term	Detail
G	Minimum Window Granularity	MSS_{min}	Minimum MSS allowed
MSS_{max}	Maximum MSS allowed (Default MSS)	MSS	Current MSS used
new_l	New measured erasure rate	E	Average erasure rate estimate
P	Number of PFEC packets in the block	R	Number of RFEC packets in the block
MSS_1	Temprorary variable	W	Congestion Window in Packets

TABLE I
TERMS USED IN THE ALGORITHM DESCRIPTION

on all SACK retransmissions being received correctly and enough dupacks and SACKs reach the source. Without reactive FEC, the retransmissions triggered by SACK would be vulnerable to losses, and any single loss of these retransmissions would trigger a timeout.

Any mechanism involves making tradeoffs. The tradeoffs of our mechanisms are as follows. Adaptive MSS uses smaller segments when windows are small and therefore the header (or packetization) overhead is larger during such phases. This overhead diminishes as window sizes grow. Proactive FEC may lead to a small deadweight goodput degradation due to over-estimation of erasure rate, and some increased burstiness in the release of dupacks from the destination. Reactive FEC triggered by each dupack leads to a somewhat increased load and burstiness in the retransmission periods. However, since these mechanisms are all adaptive (i.e., they become more active only during higher erasure rate conditions), we argue that the tradeoffs are worth making to achieve a significant gain in the dynamic range of TCP performance.

A. Detailed Scheme Description

This section fills in further detail to the earlier overview. The scheme functions are divided into the sender and the receiver modules. The sender module is responsible for adaptive MSS adjustment to provide the requisite TCP window granularity, calculating the required proactive and reactive FEC protection, and the transmission of FEC packets. The receiver implements packet reconstruction (using FEC if necessary).

The sender and receiver operations are summarized next. Explanations of the terms used can be found in Table I.

1) Sender Operation: Algorithm 1(a) provides the summary of sender operations including adaptive MSS and proactive FEC functionalities. As mentioned earlier, function $f(E)$ maps the erasure estimate, E to the minimum number of proactive FEC packets necessary. Depending on the window size in bytes, the MSS is either reduced or increased to accommodate the required number of FEC packets while providing adequate erasure protection. Thus, the variation in MSS is governed by the following factors:

- The window must be large enough to maintain 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 (see Figure 1(a) for details).
- The MSS chosen must be bounded by the MSS_{min} and MSS_{max} values.

Reactive FEC: The proactive FEC packets needed, P, specified by Algorithm of Figure 1(a) is added to the number of reactive FEC packets needed, $R = g(E)$. The sender calculates FEC such that the total redundancy packets generated are greater than $P + R$. P proactive FEC packets form a part of the congestion window while the R reactive FEC packets are used in response to duplicate acknowledgments from the receiver as described earlier.

```
P = f(E)
MSS_1 = (W x MSS x E)/P

IF(MSS_1 < MSS_min)
        W = (W x MSS)/MSS_min
        MSS = MSS_min
        IF((W x E) < 1)
                R = 1
        ELSE
                R = W x E
ELSE
        IF(MSS_1 > MSS_max)
                IF(W/MSS_max > G)
                        W = (W x MSS)/MSS_max
                        MSS = MSS_max
                        R = W x E
                ELSE
                        W = G
                        MSS = W/G
                        R = W x E
                ELSE
                        IF(W/ MSS_1 > G)
                                W = (W x MSS)/ MSS_1
                                MSS = MSS_1
                                R = P
                        ELSE
                                W = G
                                MSS = W/G
                                R = W x E
```

(a) Sender Proactive FEC Algorithm

```
IF (Data Packet)
        IF (Next in Sequence)
                IF (No Previous Unprocessed Saved Packets in the Block)
                        Process SACK Packet
                ELSE
                        IF ( (Packet in Current Block) AND (Lost Packets < P))
                                Save the Packet
                        ELSE
                                Process all Saved SACK Packets
        ELSE
                IF ( (Packet in Current Block) AND (Lost Packets < P))
                        Save the Packet
                ELSE
                        Process all Saved SACK Packets
ELSE    //Redundancy Packet
        IF (Any Saved Packets)
                IF ( (Packet in Current Block) AND (Lost Packets < P))
                        Recover and Process all SACK Packets
                ELSE
                        IF ((Last Packet in Block) OR (Packet in a New Block))
                                Process all Saved SACK Packets
                        ELSE
                                Save the Packet
```

(b) Receiver FEC Decoder

Fig. 1. Operational Steps of proactive FEC encoder implemented at the sender and the FEC decoder used by the receiver. The receiver executes Algorithm (b) for each packet arrival.

Functions $f(E)$ and $g(E)$: The function f(E) is divided in two parts:

- The first part maps the estimated erasure rate to minimum number of proactive FEC packets needed. The erasure rate is divided into multiple *bins*. Depending on the bin the estimated erasure rate falls in, we select a hard-coded number of FEC packets that define the minimum number of proactive FEC packets needed.
- Then, based on the current window size and the erasure rate, the number of proactive FEC packets used, P, is selected such that the window granularity requirements are met and P FEC packets provide sufficient protection against the estimated erasure rate, E.

Function $g(E)$ is a similar mapping like the first part of $f(E)$ where each duplicate acknowledgement leads to transmission of R reactive FEC packets depending on the erasure *bin* that E falls in.

2) Receiver Operation: Algorithm of Figure 1(a) presents the detailed operation of the FEC decoder implemented at the receiver. The decoder attempts to recover any erasures with FEC. Algorithm of Figure 1(a) is executed for each packet reception.

3) Erasure Rate Estimation: LT-TCP uses exponential weighted moving average (EWMA) to maintain an estimate of the packet erasure rate. The EWMA parameters are adaptivily biased toward higher erasure rates with

$$\alpha = \frac{new_l}{new_l + E}, \quad \beta = 1 - \alpha = \frac{E}{new_l + E} \tag{1}$$

The averaging process is performed as follows:

$$E = \alpha \times new_l + \beta \times E \tag{2}$$

While other estimation choices are possible, keeping the bias toward higher erasure rates helps in keeping the proactive FEC levels high enough to protect against small variations in average erasure rate. As shown in later in Section IV Equation 2 tracks the average erasure rate fairly well if the rate variations are small. But, as is the case with all estimates, Equation 2 tends to either overestimate or underestimate the erasure rate if the variations are bursty. Under such conditions the proactive FEC algorithm will add deadweight FEC that is either insufficient to

provide required protection or is more than the level required. Since previous studies have noted that the erasure rates are relatively stable over intervals as large as a second [1], we feel that the estimate we use will track the actual erasure rate fairly closely over most wireless channels. 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 the ACK information to do the same.

IV. Performance Results

In this section we present extensive results on the performance of LT-TCP. We also compare the performance of LT-TCP with the baseline of TCP-SACK, and two link level schemes: one which introduces link level FEC to match the packet erasure rate (PER) on the channel and another that is a hybrid FEC/ARQ scheme that is designed to provide protection for a target of 10% error rate. LT-TCP performs better than all the schemes compared, especially as the PER increases (up to 30-40%). We examine the contribution of the individual components of LT-TCP and show how they complement each other. We also observe the behavior of LT-TCP as the PER goes beyond 30% and show how the increase in timeouts reduces the absolute performance of LT-TCP. This suggests the possibility of further work to improve LT-TCP performance. The broad objectives of the performance study are as follows:

1) **Erasure Resilience:** We demonstrate LT-TCP's performance under high, variable erasure conditions compared to the baseline, which is TCP-SACK responding to both losses and erasures. A single-bottleneck test case (10 Mbps bottleneck, 20 ms one-way delay, 10 TCP flows) with erasure rates varying from 0% to 50% is used. The performance of the scheme is examined based on the end-end metrics of aggregate throughput and goodput. We use the behavior of the congestion window, number of timeouts, bottleneck queue dynamics and examination of the FEC overhead to provide a complete understanding of LT-TCP's erasure resilience. The baseline TCP version used is TCP-SACK with ECN.
2) **Performance Contribution of Components:** We illustrate the contribution of each component of LT-TCP and the residual trade-offs (ECN-only, adaptive MSS, reactive-FEC and proactive FEC). A 10% PER test case is used. We also look at the evolution of TCP's MSS and *cwnd* (in segments) over time to understand the adaptivity of MSS, its triggers and effects. We also look at the behavior of the PER estimator which controls the proactive FEC overhead we introduce.
3) **Link-Level Hybrid ARQ/FEC:** We compare LT-TCP's performance relative to two reasonable link level schemes. The first is a link level FEC scheme that uses as much FEC overhead as needed to match the PER of the channel (for the purposes of the simulation and our comparison with LT-TCP, we assume that the link has perfect knowledge of the erasure rate). We also compare LT-TCP with a reasonable link level Hybrid ARQ/FEC strategy which is designed with 10% static FEC rate with three (3) ARQ retries at the link layer (to work well at a target PER of 10%.)

A. Single Wireless Bottleneck Tests

The single-wireless bottleneck topology used is shown in Fig. 2. All end-hosts are ECN-enabled, with the bottleneck implementing RED with ECN marking. The router has a 500 packet buffer (with a nominal packet size of 500-bytes); RED $minthresh$ and $maxthresh$ values are 75 and 225 packets respectively. The simulations were run for 1000 seconds, sufficient for steady state observations. Results are averaged over 5 simulation runs, to overcome randomization effects.

Tables II and III present the performance of TCP-SACK and LT-TCP respectively. As expected TCP-SACK and LT-TCP perform well without packet erasures. However, as one would expect, the performance TCP-SACK

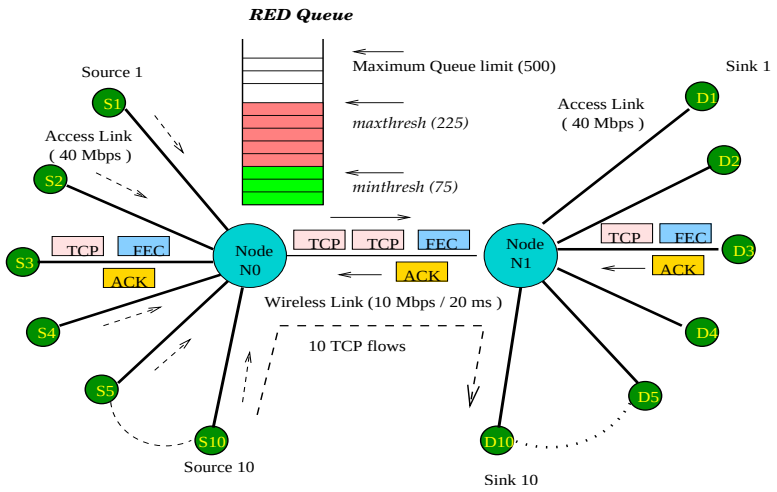


Fig. 2. Single Wireless Bottleneck Setup: RED AQM with ECN.

	ERROR RATE					
PARAMETER	0 percent	10 percent	20 percent	30 percent	40 percent	50 percent
Goodput(Mbps)	9.158	1.098	0.233	0.048	0.01	0.003807
Number of Timeouts	0	267	287	135	52	26
Throughput (Mbps)	9.52	1.272	0.306	0.073	0.018	0.007984

TABLE II

TCP-SACK W/ ERASURES: TCP GOODPUT AND THROUGHPUT, THOUGH ACCEPTABLE AT 0% PER, DRAMATICALLY REDUCES ABOVE 10% PER. FEWER TIMEOUTS AT HIGHER PER REFLECT LONGER TIME SPENT IN EACH TIMEOUT DUE TO TIMER-BACKOFF (KARN'S ALGORITHM).

drops quickly for PER of 10% and higher. LT-TCP on the other hand, shows consistent and significant relative performance improvement over TCP-SACK. LT-TCP achieves a significant goodput for all non-zero PER cases. The absolute performance (goodput) of LT-TCP is good up to about 30% PER, as shown in Table IV-A. However, for higher PER (40% and higher) the goodput drops off, while the number of timeouts goes up. The reason for the performance degradation at extremely high PER clearly lie in the sharply increased number of timeouts despite the high and adaptive FEC overhead and adaptive granulation policy. In comparison however, TCP-SACK is worse. For example even at 20% PER, TCP-SACK is operating with a very small window, as seen in Fig. 3(b) (although we recognize that TCP-SACK was not originally designed for such severe PER.) Note that TCP-SACK exhibits fewer total number of timeouts, but that is due to Karn's exponential timer back-off algorithms (triggered with back-to-back timeouts). TCP-SACK spends significantly more time in each timeout period, thus achieving very little useful goodput. We recognize the need to examine how to adapt LT-TCP to improve its ability to avoid timeouts. We believe that this approach of adjusting the overhead associated with FEC at very high PER is best suited on an end-end basis, based on the end-system's need to continue to operate over paths with such "hostile" characteristics.

Figure 3 shows the dynamics of the bottleneck queue and a comparison of TCP congestion window dynamics for 10% and 20% PER. As expected the congestion window (cwnd) graphs under these packet erasure conditions show the dramatically better performance of LT-TCP compared to that of TCP-SACK. The queue behavior illustrates that the bottleneck is underutilized with TCP-SACK. In contrast, we see a relatively fully utilized bottleneck and

	ERROR RATE					
PARAMETER	0 percent	10 percent	20 percent	30 percent	40 percent	50 percent
Goodput(Mbps)	8.94	5.36	4.086	2.99	0.89	0.3
Number of Timeouts	1	24	19	40	130	243
Throughput(Mbps)	9.53	8.55	9.01	9.06	3.53	1.74
Proactive FEC Overhead (%)	2	29	45	52	53	55
Reactive FEC Overhead (%)	0	3.7	7	11	15	17

TABLE III

LT-TCP W/ ERASURES: PERFORMANCE IS MUCH HIGHER COMPARED TO TCP-SACK. GOODPUT DEGRADATION BEYOND 30% REFLECTS INCREASED FEC OVERHEAD AND DECREASED EFFECTIVENESS IN TIMEOUT AVOIDANCE.

well-managed RED/ECN-controlled queue lengths with LT-TCP. Observe that the congestion window behavior does not directly reflect the goodput obtained by the end-system (because of the FEC and packetization overheads). However, it does reflect that we have drastically eliminated timeouts with LT-TCP over these PER regimes.

The fact that TCP-SACK stops sending packets at high loss rates results in a draining of the bottleneck queue. This results in the link being idle, and a loss of goodput (Fig. 3(c)) for a PER of 20%.In comparison, the queue behavior with LT-TCP is much better (Fig. 3(d)). Reduction in the number of timeouts and the transmission of both data and FEC packets results in a much higher utilization of the link. This translates to a much higher end-end goodput in spite of the FEC overhead.

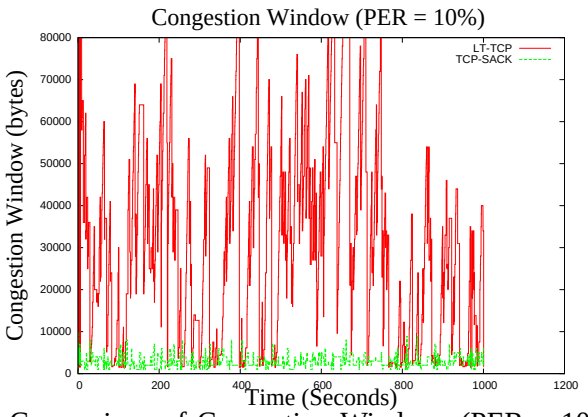
B. LT-TCP Component Performance

We evaluate the performance of LT-TCP in the single wireless bottleneck configuration (Figure 2) as we include the different individual components of the LT-TCP. This helps to understand their contribution. These experiments were with a PER of 10%, with the TCP goodput and timeout measures averaged over multiple 1000 second simulation runs with different random seeds.

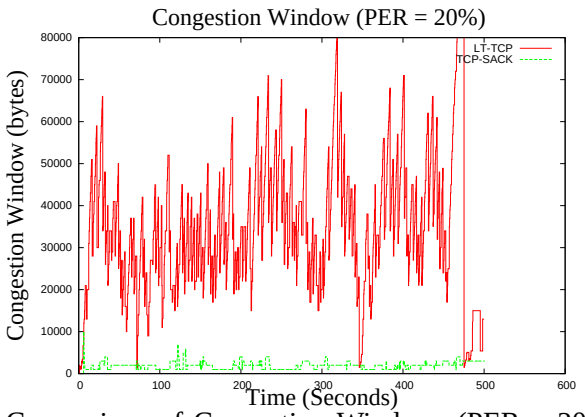
1) TCP-SACK.
2) TCP-SACK with ECN-only (i.e. RED/ECN at bottleneck and congestion response only to ECN marks).
3) TCP-SACK with ECN-only and adaptive MSS.
4) TCP-SACK with ECN-only, adaptive MSS and proactive-FEC (no reactive FEC).
5) TCP-SACK with ECN-only, adaptive MSS and reactive-FEC (no proactive 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. 4(a). The addition of each component to TCP-SACK consistently improves performance. The final goodput for LT-TCP is over five times the goodput achieved by TCP-SACK. Note that throughput (which includes FEC overhead, packetization overheads) with LT-TCP contributes to additional bottleneck link utilization which would have otherwise been wasted anyway with TCP-SACK. The residual overhead due to conservative FEC provisioning is reflected in the difference (throughput = $8.55Mbps$ vs. goodput = $5.36Mbps$). The performance gains of LT-TCP are largely explained through the reduction of timeouts.

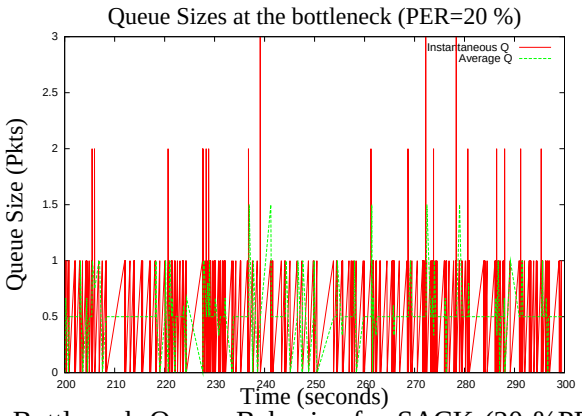
Figures 6 and 7 shows the behavior of the adaptive MSS and the resultant effect on congestion window granulation. *cwnd* (measured in segments) remains above the level of 10 segments, thus raising the number of dupacks and increasing the effectiveness of SACK when cwnd (in bytes) is small and burst losses occur. MSS also increases



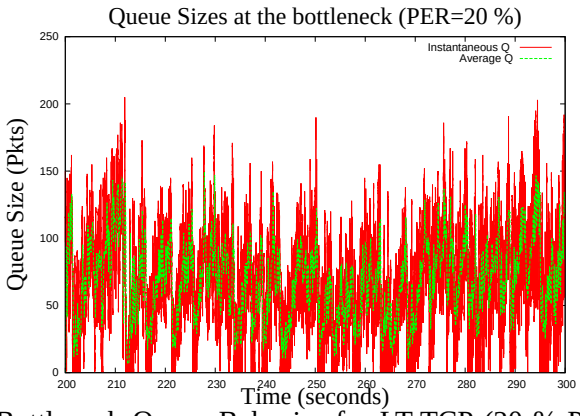
(a) Comparison of Congestion Windows (PER = 10%)



(b) Comparison of Congestion Windows (PER = 20%)

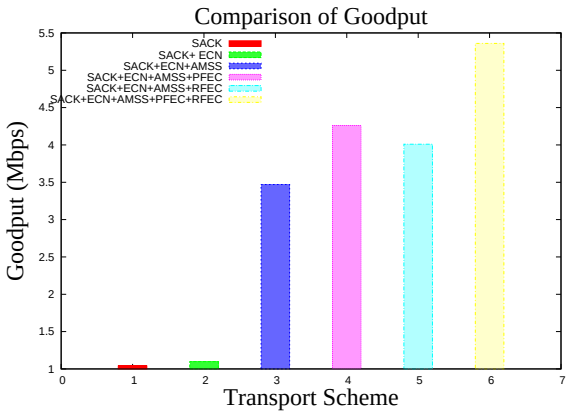


(c) Bottleneck Queue Behavior for SACK (20 %PER)

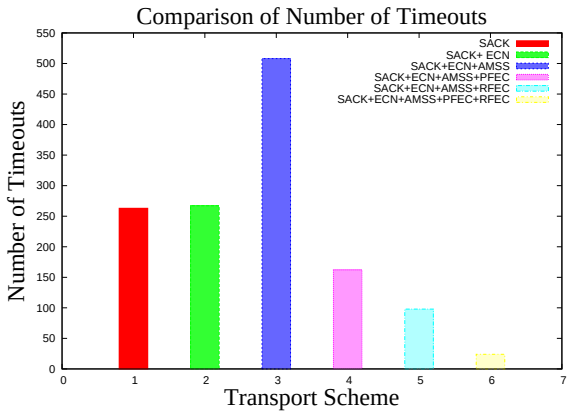


(d)Bottleneck Queue Behavior for LT-TCP (20 % PER)

Fig. 3. Graphs of Congestion Window (in bytes) and Queue Length (in packets) varying with time for TCP-SACK and LT-TCP. Results confirm LT-TCP benefits for these PERs: good bottleneck utilization and TCP throughput (partially reflects improved TCP goodput, from timeout elimination).



(a) Goodput Comparison



(b) Timeout Comparison

Fig. 4. LT-TCP Components: This figure shows the incremental contribution of each component. It can be seen that addition of additional mechanism's decreases the number of timeouts experienced. This reduction explains the bulk of performance gains.

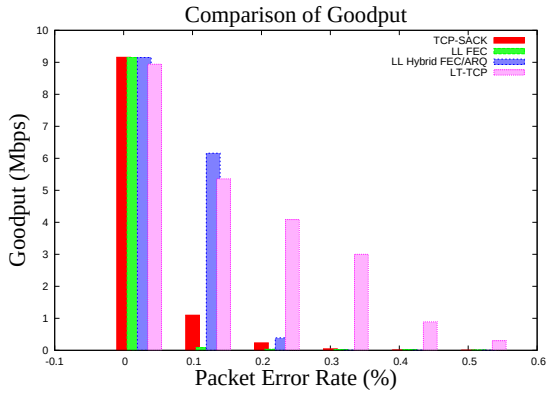


Fig. 5. Comparison of goodput for different schemes as PER varies. Only the LL Hybrid FEC/ARQ scheme does better than LT-TCP up to PER of 10 %. Beyond PER of 10%, LT-TCP performs dramatically better than the other alternatives.

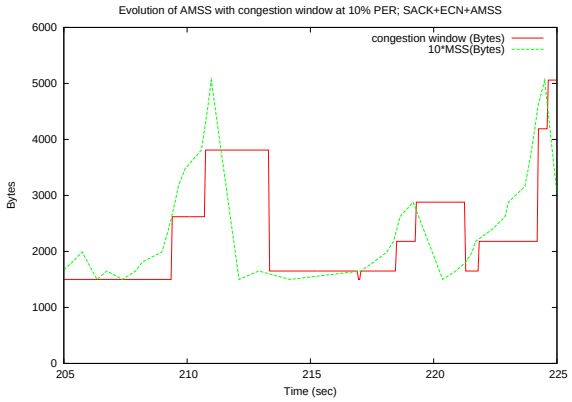
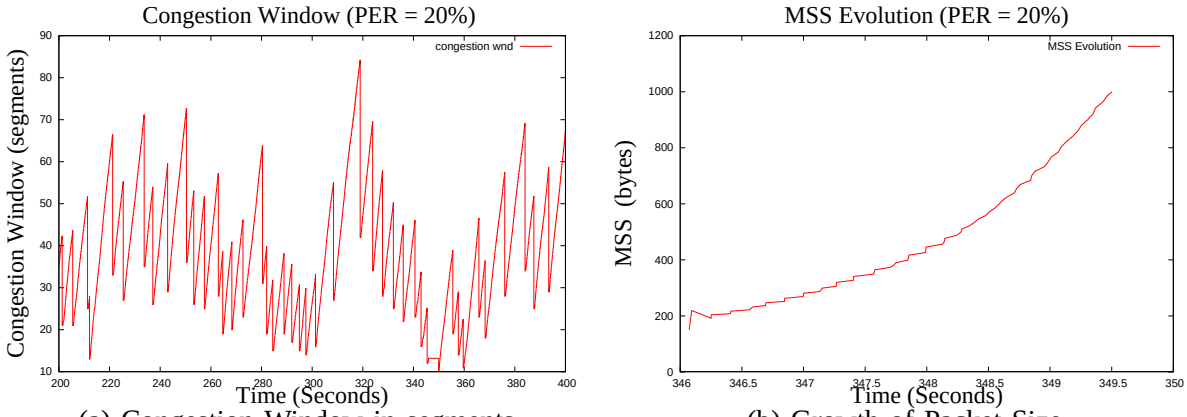


Fig. 6. Effect of Adaptive MSS: Variation of segment size (bytes) relative to the size of the congestion window (bytes) (MSS has been scaled by 10 for readability.) As the congestion window grows the packet size also increases. Enables LT-TCP to have sufficient packets to accommodate transmission of FEC packets.

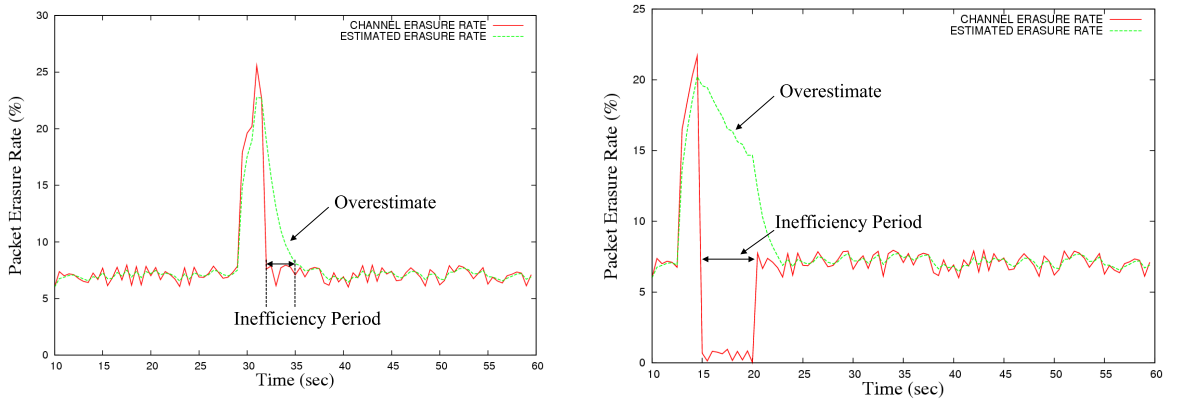


(a) Congestion Window in segments (b) Growth of Packet Size

Fig. 7. Effect of Timeout: Behavior of the congestion window and packet size following a timeout at approximately t=376 secs. The congestion window is set = 10 segments (a) while (b) shows the packet size = minimum value 150 bytes, with adaptive MSS. Growth of *cwnd* results in an increase in the segment size. Adaptive MSS ensures that the window is always greater than G segments, thus accommodating FEC packets.

when *cwnd* increases, to reduce the packetization overhead, while maintaining the minimum window granulation (G = 10). The granulation is also limited by the minimum MSS allowed.

Next we examine the behavior of the loss rate estimator and the level of proactive FEC provisioning. Fig. 8(a) and (b) shows that Equation 2 tracks the average erasure rate fairly well if the rate variations are small. But like any estimator, it tends to either overestimate or underestimate the erasure rate during a sudden bursty change in the PER. Note that our adaptive-parameter EWMA estimator (Equation 2) rapidly responds to increased erasure rate conditions, but displays a bias toward over-estimating current erasure rate conditions for a subsequent period. Prior studies have observed persistence in erasure rate conditions over periods of a few seconds [1], suggesting that this trade-off of the bias in the estimator is a reasonable one. This overestimate bias (in most cases) will lead to dead-weight FEC that is mismatched to the required degree of protection. Although this is additional overhead, it has a potential to reduce effects of sudden erasure bursts that can otherwise lead to timeouts.



(a)Transient Response (b) Over-estimation Bias

Fig. 8. This figure shows the behavior of Adaptive-Parameter EWMA erasure estimator under different conditions. Sub-figure (a) shows how a burst loss causes a transient over-estimation in the loss rate. This period is small and the estimator catches up quickly to the actual loss. Sub-figure (b) shows how a burst loss followed by a loss-less period can cause the estimator to stay in the inefficiency period for a longer time.

C. Comparison with Link-level Schemes

Simulation results [1] for number of timeouts and throughput for the four schemes is presented in Figure 9. As stated before, timeouts play an important role in determining the goodput achieved. Reactive FEC plays an important role in reducing the number of timeouts in LT-TCP. For PER between 10% - 30 %,the number of timeouts experienced by LT-TCP is quite small compared to TCP-SACK. This readily translates to an increase in goodput. However, for PERs that are higher (40% and 50%) the number of timeouts with LT-TCP increases, thus reducing the goodput. At such high PER, only LT-TCP is able to successfully send any significant amounts of data. TCP-SACK and the link level pure FEC scheme experience multiple timeouts and thus have their timeout intervals back-off exponentially. Thus, their throughput as well as goodput reduce, as seen in Figures 5 and 9. The Link level Hybrid FEC/ARQ scheme sees very low throughputs and goodput at these high PERs as well, as we observe below.

[1]The authors would like to thank Dr. Chadi Barakat of INRIA - Sophia Antipolis for generously lending us the ns-2 source code.

1) Comparison with Link Level FEC Scheme: In this subsection, we first compare the performance of LT-TCP with a link level scheme that includes an amount of FEC overhead to perfectly match the current link PER. The link level scheme operates as follows. A packet is broken up into N fragments where K units are data units that are protected by $R = N - K$ FEC packets. Each fragment is sent independently on the link. For the purposes of simulation, the amount of FEC protection i.e. $\frac{R}{N}$ was set equal to the error rate on the channel (assuming perfect knowledge of the channel PER.) The topology used was the same as in the single wireless hop scenario. This scheme is a simplification of the scheme outlined in [15]. As we see in Fig. 5, this scheme performs quite poorly even at 10% PER. The number of timeouts experienced is still fairly high in comparison to LT-TCP, as seen in Fig. 9. Table IV-C.2 presents the full complement of results for this alternative.

2) Comparison with Link Level Hybrid FEC/ARQ Scheme: We then compared the performance of LT-TCP with the full link-level hybrid FEC/ARQ technique presented in [15] (the authors tested the scheme on links with PER up to 1%.)

For the purposes of simulation, we seek a realistic mix of ARQ persistency and FEC protection but one which does not assume perfect knowledge of the PER on the channel. This is reasonable because such schemes may not wish to adaptively set the FEC protection on a measure of the PER, and rather depend on ARQ to overcome the residual error rate. The ARQ persistency value was set at 3 (to not impact latency adversely) and the FEC protection was set at 10 % i.e. ($N = 10, K = 9$). We observe that Hybrid FEC/ARQ provides very good performance when the static FEC protection is matched to the link PER. However, when the PER exceeds the targeted values, the performance of the hybrid FEC/ARQ rapidly reduces, as seen in Fig. 5. This alternative manages timeouts better, as can be seen in Fig 9, but for the same reasons as with TCP-SACK the throughput (and thus goodput) drops rapidly for PER above the designed (10%) range of the scheme. Table IV-C.2 presents the full complement of results for this alternative.

	ERROR RATE					
PARAMETER	0 percent	10 percent	20 percent	30 percent	40 percent	50 percent
Goodput(Mbps)	9.15	6.16	0.39	0.002	0.0003	0.0001
Number of Timeouts	0	2	260	25	9	8
Throughput(Mbps)	9.52	6.44	0.49	0.005	0.001	0.0007

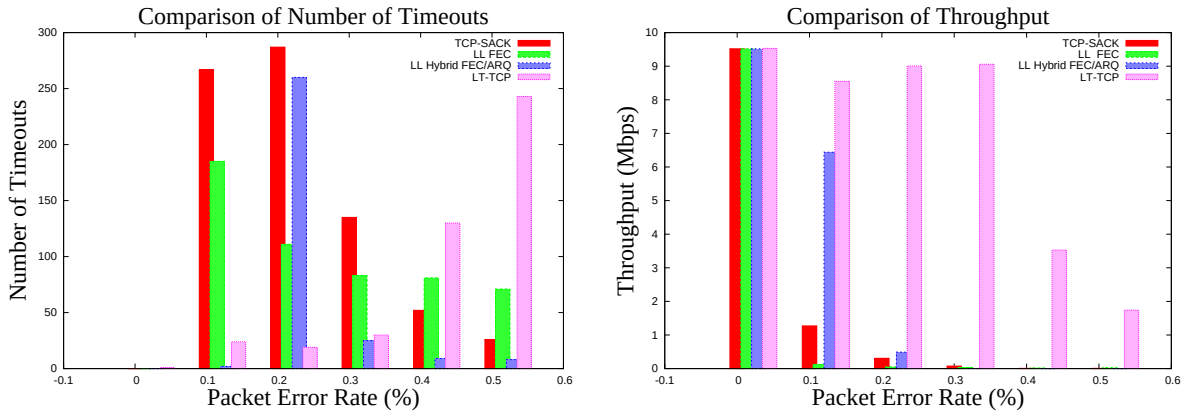
TABLE IV

SACK + HYBRID LINK LEVEL FEC AND ARQ: THROUGHPUT AND TIMEOUTS FOR DIFFERENT ERROR RATES

	ERROR RATE					
PARAMETER	0 percent	10 percent	20 percent	30 percent	40 percent	50 percent
Goodput(Mbps)	9.15	0.086	0.034	0.02	0.019	0.015
Number of Timeouts	0	185	111	83	81	71
Throughput(Mbps)	9.52	0.1235	0.053	0.03	0.019	0.026

TABLE V

SACK + LINK LEVEL FEC (FEC ASSUMING PERFECT KNOWLEDGE OF CHANNEL PER): GOODPUT, THROUGHPUT AND TIMEOUTS FOR DIFFERENT ERROR RATES



(a) Timeout Comparison (b) Throughput Comparison

Fig. 9. Comparison of timeouts and throughput (including overheads) for different schemes as PER varies. Fewer timeouts with SACK and Link Layer Schemes at higher PER is due to TCP spending longer periods in timeout, due to timer back-off. Thus, throughput (and goodput) are much lower compared to LT-TCP

V. Summary and Conclusions

This paper addressed the performance of TCP over networks that include lossy wireless links, where it is well-known that TCP performance suffers substantially when packet erasure rates (PERs) get beyond a small value of about 1% - 5%.

This paper proposed a scheme comprising a set of end-end mechanisms that substantially improve and achieve robust TCP performance in ECN-enabled networks, under extreme and highly variable erasure rate conditions. We show that TCP can still be functional with appropriate erasure protection mechanisms in networks comprising lossy wireless links even for PERs over 30%. The enhanced TCP, which we call Loss-Tolerant TCP (LT-TCP) uses FEC adaptively on an end-end basis for TCP to overcome the estimated loss rate of the channel.

LT-TCP includes a set of mechanisms that complement ECN and cooperate with each other to enhance TCP's throughput and match the amount of error protection to the conditions of the channel. We have designed LT-TCP carefully such that it introduces negligible overhead in the loss-free case, as one would demand of TCP that also needs to operate over wired links that do not have bit errors. In the case when bit errors/ packet erasures exist, the role of error protection and adaptive MSS variation by LT-TCP is to avoid retransmissions and more importantly minimize timeouts. LT-TCP does not require any additional router functionality beyond ECN (which has been standardized.) As such, it may be easily implemented on an end-to-end basis.

We compared the performance of LT-TCP with TCP-SACK and showed that for the case of 10% packet erasure rate, LT-TCP achieves a factor of 5 better TCP goodput, while introducing about 30% overhead for FEC on the channel at these error rates. We also compare LT-TCP with the contemporary approach for wireless links, which is to introduce a link-level FEC and hybrid ARQ/FEC protections scheme. We show that LT-TCP dramatically outperforms a simple link level FEC scheme, even when the link level FEC scheme adjusts its FEC overhead to exactly meet the long-term error conditions of the channel. In addition, we show that LT-TCP also outperforms link-level hybrid FEC/ARQ scheme that is designed to provide reasonable protection and performance even at 10% erasure rates. When the erasure rate goes up, the hybrid FEC/ARQ link level scheme (as well as the others) do not provide adequate protection. On the other hand, LT-TCP continues to achieve a gradual degradation in goodput until the erasure rate goes beyond even 30%. In addition, we observe that LT-TCP can complement existing link layer schemes to overcome the residual PER that may exist on the wireless channel.

We claim that LT-TCP shows consistent and significant *relative* performance improvement for all non-zero erasure rate cases in comparison to the other approaches. However, its *absolute* performance (especially goodput) suffers for very high erasure rates (40% and higher). Reasons for this clearly lie at the inability to avoid sharply increased timeouts despite the high and adaptive FEC overhead and adaptive granulation policy, and addressing this will be the focus of our immediate future work.

REFERENCES

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

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

[3] . Liu et al, "TCP-Cognizant Adaptive Forward Error Correction in Wireless Networks," ...

[4] Tal Anker, Reuven Cohen and Danny Dolev, "Transport Layer End-to-End Error Correcting," ...

[5] R. Ludwig, A.D. Joseph, "Optimizing the End-to-End Performance of Reliable Flows over Wireless Links," Proceedings of Mobicom 1999.

[6] J. Nonnenmacher and E. Biersack, "Reliable Multicast: Where to use FEC," *Protocols for High-Speed Networks*, pp. 134-148, 1996

[7] L.Rizzo, "On the feasibility of software FEC", *DEIT Technical Report*, LR-970131, Available at `http://citeseer.ist.psu.edu/rizzo97feasibility.html`

[8] C. Huitema, "The Case for Packet Level FEC", *Proc. of IFIP 5th Int'l Workshop on Protocols for High Speed Networks*, October 1996.

[9] Y. Yang, H. Zhang and R. Kravets, "Channel Quality Based Adaptation of TCP with Loss Discrimination", *Proceedings of IEEE GLOBECOM*, November 2002.

[10] S. Dawkins, G. Montenegro, M. Kojo, V. Magret and N. Vaidya, "End-to-end Performance Implications of Links with Errors", *IETF RFC 3155*, August 2001.

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

[12] G. Fairhurst and L. Wood, "Advice to link designers on link Automatic Repeat reQues (ARQ)",*IETF RFC 3366*, August 2002.

[13] R. Ludwig and M. Meyer, "The Eifel Detection Algorithm for TCP", *IETF RFC 3522*, April 2003.

[14] C. Barakat and E. Altman, " Bandwidth tradeoff between TCP and link-level FEC", *Computer Networks*, vol. 39, no. 5, pp. 133-150, June 2002.

[15] C. Barakat and A. A. Fawal, " Analysis of link-level hybrid FEC/ARQ-SR for wireless links and long-lived TCP traffic", *Performance Evaluation Journal*, vol. 57, no. 4, pp. 423-500, August 2004.

[16] B. Liu, D. Goeckel, D. Towsley, "TCP-cognizant adaptive forward error correction in wireless networks", *Proc. IEEE Globecom*, November 2002,.

[17] S. Biaz and N. H. Vaidya, "Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver", *IEEE Symposium ASSET'99,Richardson*, March 1999.

[18] S. Biaz and N. H. Vaidya, "Distinguishing Congestion Losses from Wireless Transmission Losses", *Seventh International Conference on Computer Communications and Networks (IC3N)*, October 1998.

[19] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", *Proceedings of ACM Mobicom 2001*, pp 287-297, July 2001

[20] S. Cen, P. C. Cosman, and G. M. Voelker, "End-to-End Differentiation of Congestion and Wireless Losses", IEEE/ACM Transactions on Networking, Vol.11, No. 5, October, 2003.

[21] Cisco Inc., "THE CISCO POSITION ON WIMAX AND RELATED NEXT-GENERATION RADIO TECHNOLOGIES FOR MOBILE OPERATORS", 2005.
`http://www.cisco.com/en/US/netsol/ns341/ns396/ns177/networking/solutions/white/paper0900aecd801aa448.shtml`

[22] R. Krishnan, J. P. G. Sterbenz, W. M. Eddy, C. Partridge and M. Allman, "Explicit Transport Error Notification (ETEN) for Error-Prone Wireless and Satellite Networks",*Computer Networks*, Volume 46 , Issue 3, pp. 343 - 362, October 2004.

[23] H. Balakrishnan and R. Katz, "Explicit Loss Notification and Wireless Web Performance", *IEEE GLOBECOM*, November 1998.

[24] K.K. Ramakrishnan, S. Floyd, and S. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", *IETF RFC 3168*, September 2001.

[25] S. Dawkins, G. Montenegro, M . Kojo and V. Magret, ”End-to-end Performance Implications of Slow Links”, *IETF RFC 3150*, July 2001.

[26] B. Skalar, ”Digital Communications, Fundamentals and Applications”, 2nd Edition, Prentice Hall, 2001.

[27] I. Reed and G. Solomon, ”Polynomial codes over certain finite fields,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, pp. 300-304, June 1960.

[28] M. Luby, ”LT- codes”,*Proceedings of the 43rd Annual IEEE Symposium on the Foundations of Computer Science (STOC)*, pp. 271-280, November 2002.

[29] M. Allman, D. Glover and L. Sanchez, ”Enhancing TCP Over Satellite Channels using Standard Mechanisms”, *IETF RFC 2488*, January 1999.

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

[31] H. Balakrishnan, S. Seshan, E. Amir and R. H. Katz, ”Improving TCP/IP Performance over Wireless Networks”, *ACM Conf. on Mobile Computing and Networking*, November 1995.

[32] Mun Choon Chan, Ramachandran Ramjee, ”TCP/IP performance over 3G wireless links with rate and delay variation” *Mobicom*, pp. 71-82, September 2002.

[33] ”Performance Implications of Link Characteristics (PILC)”, `http://www.isi.edu/pilc/`.

[34] R. Ludwig, A. Konrad and A. D. Joseph, ”Optimizing the End-to-End Performance of Reliable Flows over Wireless Links”,*MobiCom* , August, 1999.

[35] M. Mellia, C. Casetti and Michela Meo, ”TCP Smart Framing: Using Smart Segments to Enhance the Performance of TCP”,*IEEE Globecom*, November 2001.

[36] L. J. Chen, R. Kapoor, M. Y. Sanadidi and M. Gerla, ”Enhancing Bluetooth TCP Throughput via Link Layer Packet Adaptation,” *IEEE ICC*, June 2004.