TRANSPORT AND LINK-LEVEL PROTOCOLS FOR WIRELESS NETWORKS AND EXTREME ENVIRONMENTS

By

Vijaynarayanan Subramanian

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Computer and Systems Engineering

Approved by the
Examining Committee:

Shivkumar Kalyanaraman, Thesis Adviser

K.K. Ramakrishnan, Member

Koushik Kar, Member

Kenneth Vastola, Member

Christopher Carothers, Member

Rensselaer Polytechnic Institute
Troy, New York

April 2008
(For Graduation May 2008)

TRANSPORT AND LINK-LEVEL PROTOCOLS FOR WIRELESS NETWORKS AND EXTREME ENVIRONMENTS

By

Vijaynarayanan Subramanian

An Abstract of a Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Computer and Systems Engineering

The original of the complete thesis is on file
in the Rensselaer Polytechnic Institute Library

Examining Committee:

Shivkumar Kalyanaraman, Thesis Adviser
K.K. Ramakrishnan, Member
Koushik Kar, Member
Kenneth Vastola, Member
Christopher Carothers, Member

Rensselaer Polytechnic Institute
Troy, New York

April 2008
(For Graduation May 2008)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

I would like to thank my thesis adviser Prof. Shivkumar Kalyanaraman for the guidance he has given over the past few years. Apart from his technical expertise, I greatly appreciate his patient and encouraging attitude and the freedom he gives to his students.

I am grateful to Dr K.K.Ramakrishnan of AT&T Labs (Research) for being an unofficial co-adviser for my thesis. My stints as an intern under him were valuable learning experiences.

Many thanks also to my other committee members Prof. Koushik Kar, Prof. Ken Vastola and Prof. Chris Carothers for agreeing to be on my committee.

ABSTRACT

The widespread deployment of wireless links and the expected explosion in the use of wireless broadband systems makes it important for link and transport layer protocols to perform well in these environments. Wireless paths, especially multi-hop paths present challenges in terms of bounded end-to-end delay, end-to-end packet error rate (PER) experienced by the transport layer and achievable goodput. Current transport protocols such as TCP-SACK are known to fail as the PER goes above 5 % PER.

In this thesis, we develop transport and link-level protocols that use the principles of fragmentation, loss estimation and Forward Error Correction (FEC). Our design structures these building blocks to perform well even under high loss rates (up to 50 % PER). Analysis of the scheme enables us to tune the protocols based on their different needs while providing insight into the most favorable configuration. We also illuminate the issue of balance of functionality and interactions between the link and transport layers. At the link layer, these mechanisms are designed to meet the objectives of having bounded delay while exporting a negligible residual error rate. Link-layer mechanisms are designed to be robust under conditions of disruptions when no communication is possible. This is accomplished by using mode-switching and outage detection techniques designed to combat disruptions. However, over multi-hop paths, per-hop residual loss rates may aggregate and result in significant end-to-end loss rates (¿ 5%). Link-level mechanisms by themselves are insufficient and support at the transport layer is needed under high end-to-end loss rate scenarios. At the transport layer, we concentrate on tackling the residual error rate while operating on a longer time-scale. Our contributions include: (a) A highly loss-tolerant TCP protocol (LT-TCP) using an adaptive, end-to-end hybrid ARQ/FEC reliability strategy exploiting ECN for congestion detection. (b) A link-layer hybrid ARQ/FEC protocol (LL-HARQ) resulting in small residual PER, bounded link latency and high goodput even under bursty/high loss and outage conditions. (c) Demonstration that the combination achieves improved end-to-end

performance (delay, loss and goodput) over traditional approaches. We present performance results for a comprehensive set of scenarios including different loss/error models as well as under different (1-hop and multi-hop) topologies.

We then tested the LT-TCP protocol in an 802.11b setting where we consider errors introduced by cross-system interference (by modeling Bluetooth traffic) as well as co-channel interference (by modeling two interfering 802.11 cells). We find that such setups are susceptible to "capture" or channel outages and that LT-TCP can provide benefits under such capture scenarios, particularly as the RTT increases. Measurements on the Open-Access Research Testbed for Next-Generation Wireless Networks (ORBIT) were used to study the impact of noise-induced errors on the performance of currently deployed protocols in real systems. We show how the performance of current protocols degrades as the error rate go up and how retransmissions at the link layer can lead to unwanted interactions between link and transport layers. These measurements give insights into the impact of wireless packet losses and bolster the case for developing protocols that are robust to high loss rates. Finally, we used real-world traces gathered from airplane flights to study the nature of outage and disruption events and to model packet losses on airborne wireless links. These link characteristics and models of packet losses were used to test and refine our link and transport protocols and to validate the efficacy of our proposed solutions under realistic conditions.

CHAPTER 1
Introduction

Wireless network deployments have grown dramatically over the last few years (Figure 1.1). The rapid deployment of broadband wireless systems such as 802.11 wireless local area networks (WLANs), 802.16 wireless broadband and neighborhood area wireless networks raises expectations of high end-to-end performance. Figure 1.2 from [13] shows wireless network deployment in an area of Boston, USA. The high density of these wireless networks raises concerns about potential interference among these wireless cells. Such deployments in open spectra (e.g. in ISM band) will only grow as users expect higher performance (in terms of latency and goodput) and services over wireless systems.

It is anticipated that future cells will be small in order to deliver high bandwidth to users. This leads to interference among cells in close proximity. There exists anecdotal evidence of such co-channel interference in apartment complexes in dense urban areas such as Singapore when access points are set up independently, without global planning or coordination. This phenomena is already apparent in densely populated residential areas, flats and apartments where any client can see a large number of access points, and can interfere with their transmissions during peak usage periods even if the client cannot make an association with the access points (i.e., when access control is enabled). Moreover, wireless networks using the same frequency near one another will lead to not only interference but also disruption phenomenon where all packets sent on the channel are corrupted.

Thus, wireless systems operating in open-spectrum bands will be used in environments characterized by higher levels of interference, capture and disruption phenomena. Communication under these demanding conditions will require link and transport layer protocols to handle not only medium access and packet errors, but also handle unexpected periods of capture and disruption due to unavoidable interference. WLAN cells may be vulnerable to capture effects that occur due to interference and inability of the MAC layer to fairly distribute transmission opportu-

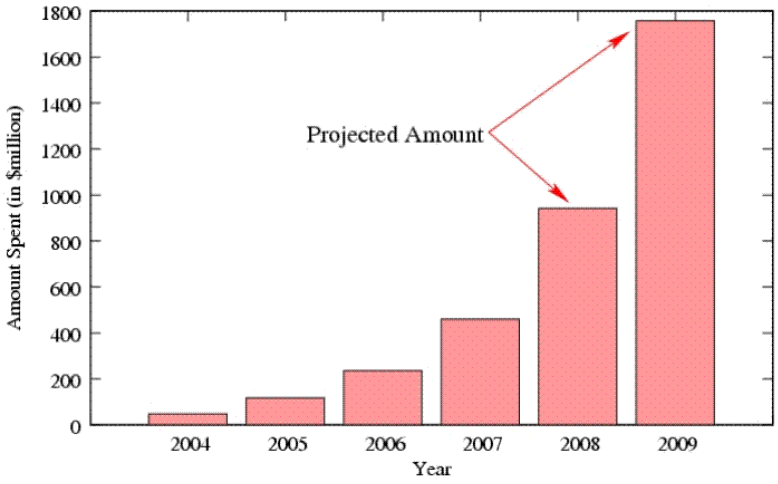


Figure 1.1: This figure shows the past investments in wireless deployments and the projections in the near future.

nities. Disruptions or *outages* of several hundred milliseconds (where all packets are lost) are possible under such conditions, especially during peak usage hours. Current link layer protocols are unable to detect and react to such prolonged disruption periods. We demonstrate this issue in the context of 802.11 LANs experiencing co-channel interference in chapter 6.

This widespread deployment of wireless links, wireless broadband systems and the emergence of multi-hop wireless networks poses challenges to protocols such as Transmission Control Protocol (TCP) because of the potential for high residual loss rates. TCP is the *de facto* workhorse transport protocol of the Internet that was originally designed in an era of low bandwidth and congestive losses. As applications are ported to wireless environments, it is reasonable to assume that they will continue to use TCP as the transport layer protocol.

As the demand for broadband connectivity increases, both cellular and meshed networks will play a role in last-mile wireless distribution networks. Current MetroWiFi deployments (e.g. San Francisco, Taipei etc..) and organic community wireless deployments fit this model. While the performance of TCP/IP has been well studied for one-hop cellular-style low-speed wireless networks, TCP performance over higher bit-rate, lossy, multi-hop wireless networks is not well understood. It is well-known that wireless links have high, bursty and variable raw error rates due to atmospheric

Figure 1.2: This figure shows wireless deployments in the city of Boston, USA. The high concentration and density of APs can be seen clearly[13].

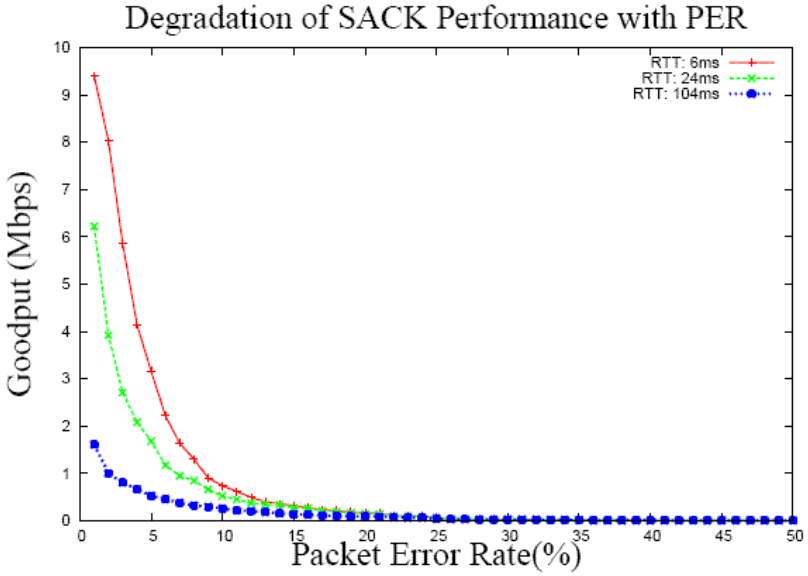


Figure 1.3: TCP-SACK Degradation with Increased Erasure Rate and RTT (Uniform Loss Probabilities, 10 Mb/s Capacity, 1 flow)

conditions, terrestrial obstructions, fast and multi-path fading, active interference and mobility[16]. However, for TCP, what matters is residual packet erasures and delay behavior after PHY and LINK layer mitigation has been completed [50]. TCP is exposed to residual error rates which is defined as the error rate subsequent to the link layer's error protection mechanisms.

The performance of state-of-the-art TCP variants such as TCP-SACK deteriorates badly as the end-to-end loss rate increases, primarily as the result of mis-interpreting wireless packet losses for congestion. TCP was designed for networks where the probability of packet corruption was negligible (below 1%) and the assumption that a packet loss signaled congestion was valid. In such a scenario, reducing the transmission rate was the correct response. In wireless environments however, packet losses due to packet corruption is common. Cutting the transmission rate in such a scenario is therefore counter-productive. This well-known problem causes TCP performance to degrade on wireless links due to packet corruption being misinterpreted as congestion losses. Figure 1.3 shows the performance of TCP-SACK as the packet loss rates and round-trip-times (RTTs) are varied. It can be seen that the degradation in performance is rapid and that an end-to-end loss rate of around 5% is sufficient to cause the connection to collapse.

Multi-hop wireless networks are rapidly emerging as an integral means of data communication, complementing wired networks. Such networks include mobile ad-hoc networks (MANETs) and meshed wireless networks. MANETs in turn find expression in scenarios such as vehicular *ad hoc* networks (VANETs), home networking, networks of laptops or PDAs, and multi-tier tactical battle-space networks (ground-based networks on vehicles and soldiers and airborne networks of UAVs (see Figure 1.4). Meshed networks may also be found in a managed context (WiMax back-haul, homeland security applications, Nokia Rooftop) as well as in largely unmanaged community wireless networks.

Multi-hop paths also experience local path disruptions and re-establishment that impact TCP [33], but we focus on the end-to-end behavior when the path is relatively stable (or may change locally), but is lossy. Unlike wired inter-networks, the end-to-end path in a multi-hop wireless environment may have a large number

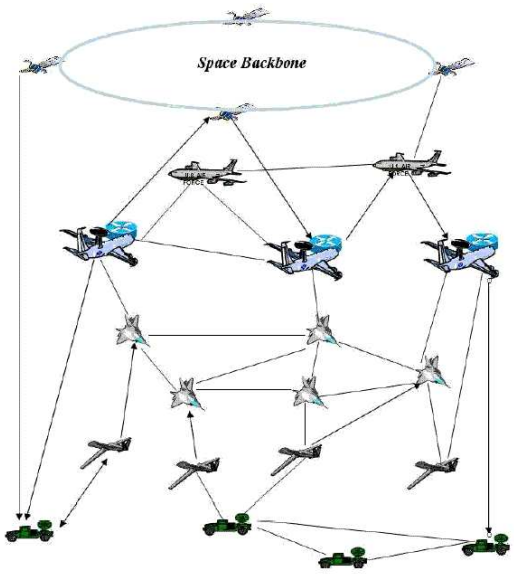


Figure 1.4: The figure depicts a possible battlefield scenario wherein communication over multiple wireless hops may be desired under stressful conditions.

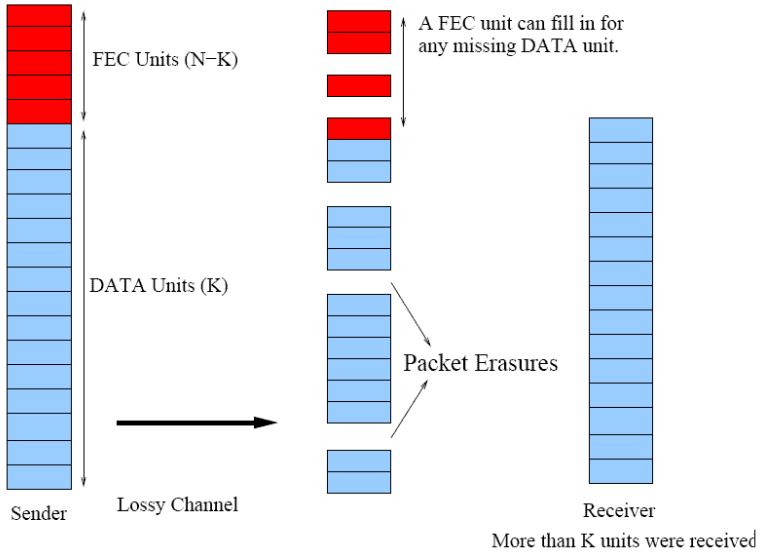


Figure 1.5: Operation of Forward Error Correction: RS coding has the property that any FEC unit can stand in for a lost data unit. As long as K out of N units are received, the original data bytes can be recovered.

of hops that are vulnerable to disruptions over multiple time-scales (e.g., bit/packet errors, burst errors, interference lasting multiple round-trip times, etc.). This makes it challenging to achieve good performance in terms of end-to-end bounded latency, packet error/loss rate (PER) and achievable goodput. We believe it has become necessary to enhance transport protocols such as TCP to work in these challenging environments where end-to-end loss rates can be significant.

Various mechanisms have been employed at the link-layer to minimize the effect of interference and noise with the aim of reducing the loss rate seen by the transport layer. Automatic Repeat Request (ARQ) and Forward Error Correction (FEC) and combinations of these (Hybrid schemes) are popular approaches used to reduce the residual error rate on the link. An illustration of Forward Error Correction is shown in Figure 1.5. FEC is discussed in more detail in chapter 2.

Despite incremental improvements to TCP design and implementations, it cannot operate efficiently over high bandwidth-delay product paths, over links with high packet erasure rates and cannot leverage the availability of multiple end-to-end paths. Multi-path LT-TCP is an emerging idea that focuses on obtaining diversity gains through the use of multiple-paths between the sender and receiver. This will enable robustness particularly under conditions of large time-scale error conditions. In this thesis, we focus only on a single-path connection.

We also study the problems of congestion control and reliability. We propose end-to-end TCP upgrades that can work without any link or network support (bar Explicit Congestion Notification (ECN)), and can leverage any available network support. There are several proposals for incorporating reliability mechanisms at the PHY, LINK or transport layers, which work well under specific circumstances [9, 25]. However, we are interested in a general purpose solution that works well for a broad range of applications and dynamically adapts to the underlying network conditions. We show that it may not be wise to expect a single layer (link or transport) to provide such a complete reliability solution. Therefore, we investigate the problem of finding the right balance between mechanisms and functions at the various layers to improve overall performance.

Link-level protocols attempt to mitigate the effects of lossy environments. Tra-

ditionally, the view of the link-level designers has been to provide a low-loss link by means of multiple ARQs (e.g., IEEE 802.11) and PHY level Forward Error Correction (FEC). While link protocols with persistent ARQ capability can make the link layer reliable, there exists a clear trade-off between latency and reliability. However, as the number of hops increase, the latency penalty also grows. This penalty is exacerbated particularly when link-layer protocols back-off between successive re-transmission attempts.

In summary, we see that the emerging rapid deployment of high data rate, multi-hop wireless networks cause significant problems to existing link and transpot protocols. Unplanned deployments in open-access spectra will also give rise to high raw loss rate on the link which will also cause disruptions and outages to nodes. In such *extreme* environments, to obtain a favorable trade-off at the link-layer between goodput, residual loss rate and latency, and to make the transport layer robust to high residual loss rates, we need robust link and transport protocol designs that can meet the objectives outlined below.

1.1 Research Objectives

In this work, we study and propose mechanisms both at the transport and link layers. While our mechanisms have commonalities at these layers, they are nevertheless designed to meet different objectives at the two layers. We identify the goals at the two layers.

1.1.1 Link-layer Objectives:

Our objectives at the link-layer are to provide a link with a low residual loss rate (subsequent to the link layer's error protection mechanisms), while achieving low latency and high goodput. Clearly, the goal of providing bounded and small link-level delay can be achieved only by limiting the number of ARQ retries. In this context, we pose the question "Can link-layer protocols with limited ARQ attempts (possibly one) still provide a sufficiently small residual loss rate and high goodput?". Moreover, links can be exposed to periods where the PER is 100% i.e. packets are lost with certainty. Events such as jamming, cross-channel interference and co-

channel interference may lead to such scenarios. We are also interested in developing mechanisms that can be robust under such conditions of disruptions while meeting our desired link-layer objectives.

In this work, we revisit the issue of link-level design with the objective of making it work well under disruption channel conditions. In particular, we develop link-layer disruption detection, and respond with a more conservative mode of link-layer hybrid FEC/ARQ scheme (HARQ) and control the residual error rate exported to the transport layer. We define disruption as conditions under which a packet (and all of its fragments when it is fragmented at the link layer) is lost with certainty. We list below the properties that a link-level design should possess. We develop our mechanisms in the context of a generic link layer that suffers from a high packet error rate including disruptions. These methods can then be tailored and integrated within the context of specific MAC layers. Our link-layer objectives include:

Delay Control: Unbounded number of ARQ-style retransmission attempts on the link-level is not desirable from an end-to-end point of view. Link level mechanisms must be able to provide robustness while keeping the link-level latency small. This limits the number of permissible ARQ retransmission attempts.

Low Residual Loss Rate: Residual loss/error rate, as defined above should be as low as possible so that upper layers such as TCP are exposed to as small an error rate as possible. This must be done however, while keeping the overhead and link latency small.

High Link-Level Goodput: Residual loss rate can be driven to zero, either by trading off latency (more ARQ attempts) or by reducing goodput (by adding FEC indiscriminately). The link HARQ scheme must manage this trade-off because the link-level goodput (discounted by residual loss rate) puts an upper bound on end-to-end goodput.

In-order Delivery: The link-layer should attempt to deliver successfully transmitted packets in-order since delivery of packets out-of-order to protocols such as TCP will cause unintended side-effects (such as fast recovery).

Limited Impact on Transport Layer: The link level mechanisms should interact minimally with the transport layer to avoid effects such as spurious timeouts,

variable end-to-end RTT and bandwidth. Ideally, the link should provide TCP with the illusion of a constant bandwidth, zero-loss pipe. It is challenging to provide this abstraction in multi-hop, lossy and disruption-prone environments.

Robustness under Disruption: The link-level mechanisms must be able to perform well even when conditions of disruptions or outage (where no communication i.e. PER is 100%) occur.

We study the following issues.

1. How much effort should be expended in making the link-layer as reliable as possible?

2. What should be the balance between the protections offered by the link and transport layers?

3. How can this balance be achieved dynamically? i.e. Can the protocols find a favorable operating point to provide good performance?

1.1.2 Transport-layer Objectives:

Over multihop-wireless paths, link-level support alone may be insufficient to reduce the end-to-end loss rate to negligible levels (as desired by traditional reliable transports such as TCP). While link-level protocols can reduce the residual loss rate on each hop, these loss rates accumulate over multiple hops and present TCP with a significant end-to-end loss rate to overcome. In the context of reliable transport protocols, we ask "Can TCP be enhanced to operate efficiently over paths that suffer from high losses?"

Our transport-layer objectives include:

Increased Dynamic Range: Our main objective at the transport-layer is to increase the operating range of TCP so that it can perform robustly under end-to-end packet error rates ranging from 0% PER to 50% PER.

Timeout Avoidance: Since timeouts are truly wasteful (they cause the congestion window to be reset), and high error rates lead to more timeouts, an important requirement to realize high goodput under high-loss conditions is to minimize the number of timeouts experienced by TCP.

Congestion Response: Since TCP's assumption that all packet loss is due to congestion is no longer valid, an appropriate mechanism for disambiguating between congestion and wireless losses is needed. We also need to develop appropriate responses to these signals.

We answer both of the questions above about the link and transport protocols in the affirmative and present a generic framework using the principles of adaptive granulation, loss estimation, and Forward Error Correction (FEC) that enables operation over a channel suffering from high/bursty loss conditions. We then apply these principles at the link and transport layers and tune them as per their specific requirements. While the proposed protocols at the link and transport contribute to improved performance at their respective layers independently, the combination of the two schemes yields synergistic benefits that improve performance dramatically, especially under high and bursty loss regimes.

1.2 Research Contributions:

This thesis contributes to both the transport and link layers. We enumerate our contributions at the two layers below.

1.2.1 Link-layer Contributions:

The contributions at the link-layer include:

1. Providing insight into the nature of disruptions and outages on the link-level and impact on the transport protocol.

2. Proposing and evaluating a dynamic link-level design that can overcome the challenges of operating under such conditions while keeping latency and residual loss rate low.

3. Investigating the trade-offs between link-level ARQ persistence and transport level goodput. Moreover, a design goal is to develop a scheme that will work well on all types of wireless links and not just specific systems such as 802.11 links or satellite links.

1.2.2 Transport-layer Contributions:

The transport layer contributions include:

1. Dynamic maximum segment size (MSS) algorithms that limit the impact of packet erasures, and support end-to-end strategies.

2. Robust behavior under high and bursty loss scenarios characterized by high end-to-end packet error rate (PER) thereby increasing the dynamic operating range of the transport protocol.

3. Higher throughput and goodput on an end-to-end basis realized by a reduction in the number of timeouts.

4. End-to-end FEC to complement end-to-end ARQ schemes (TCP-SACK).

5. New end-to-end congestion control algorithms that offer high utilization, fairness and robust steady state behavior.

In summary, our contributions include: (a) A highly loss-tolerant TCP protocol (LT-TCP) using an adaptive, end-to-end hybrid ARQ/FEC reliability strategy exploiting ECN for congestion detection. (b) A link-layer hybrid ARQ/FEC protocol (LL-HARQ) resulting in small residual PER and high goodput even under bursty/high loss conditions. (c) Demonstration that the combination achieves improved end-to-end performance (delay, loss and goodput) over traditional approaches.

Our proposed approach is comprehensive and has a number of unique interlocking elements for high packet erasure conditions. Our solution provides benefits stemming from a design based on fundamentals but provides enough flexibility to incorporate the best of those advances. The rest of the thesis is organized as follows. Chapter 2 provides an overview of the related work. Chapter 3 provides an overview of the general design using an abstract model of a lossy channel and analyzes the components of the design and provides insight into the structuring of the various building blocks. Chapters 4 and 5 present the details of the transport and link protocols which are tuned based on the insights obtained from the analysis. Chapter 6 studies the performance of the proposed transport enhancements in IEEE 802.11

WLAN networks. Chapter 7 presents the performance evaluation of the proposed schemes using the ns-2 simulator. Chapter 8 uses real world traces from airplane flights from experiments conducted by MIT- Lincoln Labs to study airborne links and see how our proposed solutions can help in these conditions. Chapter 9 presents the results of measurements conducted on the ORBIT wireless testbed at Rutgers University to study the impact of noise-induced errors on the link and transport layers and the interaactions between the two layers. Chapter 10 summarizes and concludes the thesis and discusses future directions.

CHAPTER 2
Related Work

Transmission Control Protocol (TCP) has been the dominant transport layer protocol for several years. While problems experienced by TCP in wireless environments have been known for some time, it is the recent explosion in wireless technologies and the impending explosion in wireless-system deployments that has provided the impetus for this work. Link-layer mechanisms have also been studied and analyzed extensively. ARQ and FEC mechanisms have been in use for some time. However, the current focus on multi-hop wireless networks such as *ad hoc* networks, sensor networks and mesh networks have forced us to rethink link-layer mechanisms to meet the objectives presented in chapter 1. The menu of error control building blocks is relatively well known: a mix of FEC, ARQ, pipelining, interleaving, packetization, and adaptive modulation/coding (AMC) mechanisms [29, 145]. We submit that the open issues for emerging multihop and meshed networks operating in extreme and unmanaged environments include: a) how to structure these building blocks into protocols to achieve attractive trade-offs, b) how to divide and balance responsibilities among layers (PHY / LINK / transport) to survive a wide range of error characteristics for a broad range of applications, and (c) to understand and minimize the need for cross-layer interactions. In this chapter, we look at some of the more prominent solutions for both the transport and link-layers. We look at the enhancements made to TCP to enable it to operate over paths with high loss rates. We also look at the technique of Forward Error Correction (FEC), especially Reed-Solomon (RS) codes[119] and rateless codes[93]. The rest of the chapter is as follows. Section2.1 discusses the problems experienced by wireless systems (interference and disruptions). Section 2.2 discusses link-level proposals designed to mitigate wireless errors. Section 2.3 discusses TCP mechanisms of interest and solutions at the transport layer for wireless scenarios including those designed for multi-hop environments. Section 2.4 provides an overview of the Forward Error Correction codes of interest in this thesis.

2.1 Wireless Interference and Disruption

Multi-hop wireless networks pose challenges because of their lossy nature. Wireless networks are prone to disruptions over multiple time-scales: bursty bit errors and packet loss (small time-scale), interference and capture effects (medium time-scale) and long-term path disruptions due to persistent channel impairments and mobility (large time-scale) [64]. 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 [51]. 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 [16]. A performance study on long-range 802.11 wireless links [47] showed that the link packet error rate can vary rapidly between 0% and 100% when the received signalto-noise ratio (SNR) changes by as little as 4-6 dB. Preliminary industry studies of WiMax and mobile/portable broadband access also suggest significant variability in performance [15]. 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. Recent studies have examined the impact of interference in wireless LAN environments. Golmie et al. study the performance of Bluetooth Access Control Layer in [67] operating in close proximity to an 802.11 WLAN system. The probability of collision between a Bluetooth transmission and WLAN transmission is derived and is found to be significant. Golmie et al. [65] evaluate the effect of mutual interference on the performance of Bluetooth and IEEE 802.11b systems. The authors report significant packet error rates for WLAN transmissions given interference from Bluetooth. Shellhammer [130] derives the probability of an 802.11 packet error in the presence of interference from Bluetooth. Chiasserini et al. [49] present a model of the interference that IEEE 802.11 transmissions may experience because of either a Bluetooth call or voice link. The paper also proposes a traffic shaping technique to the Bluetooth flow that can

reduce the impact of interference. Techniques have been developed for mitigating the effects of low signal strength caused due to path-loss, shadowing and noise. Since there is a direct relationship between the signal strength and achievable data rate (required receiver sensitivity to obtain higher data rate is higher), one solution is to reduce the operating data rate when the signal strength deteriorates. Rate adaptation is a technique used by 802.11 a/b/g wireless devices to make use of multi-rate capabilities in response to SNR degradation and packet erasures. Sadeghi et al. [127, 126] discuss the Auto Rate Fallback (ARF) protocol and introduce the Opportunistic Auto Rate (OAR) protocol to better exploit link conditions. Lacage et al. [87] propose an Adaptive Auto Rate Feedback (AARF) algorithm for low latency systems. Holland et al. [34] present a rate adaptive MAC protocol called the Receiver-Based AutoRate (RBAR) protocol where the adaptivity is determined by the receiver and not the sender.

2.2 Link-layer Enhancements

It is well-known that wireless links have high, bursty and variable raw error rates due to atmospheric conditions, terrestrial obstructions, fast and multi-path fading, active interference and mobility [16, 35]. However, for TCP, what matters is residual packet erasures and delay behavior after PHY and LINK layer mitigation has been completed [51]. We look at enhancements at the link and PHY layer in this section. There is also considerable previous work on sophisticated link-layer techniques. One may put them in two broad categories: a) Hybrid FEC/ARQ [27] and b) TCP performance-enhancing-proxies (PEPs) [39, 25, 45].

2.2.1 Hybrid FEC/ARQ (HARQ) Schemes

FEC is a popular error mitigation technique for digital voice communications [133, 29]. Bit error correction is usually performed using convolution codes, Turbo codes or a mix of coding and modulation [133]. Attractive building blocks for packet erasure correction include Reed-Solomon (RS) codes [119, 145, 29] and recently-proposed rateless codes [93, 131, 41]. Unlike FEC, Automatic Repeat reQuest (ARQ) is a closed-loop mechanism that requires error/erasure detection,

feedback and retransmission [29, 147]. While FEC suffers from dead-weight overheads in benign conditions, ARQ suffers from delays that worsen in extremely lossy conditions[88]. Hybrids of ARQ and FEC (HARQ) can be created by using FEC in the retransmission phase (e.g. using rate-compatible punctured convolution codes (RCPC), RS-codes [119] or Turbo codes[133]), and by jointly decoding over all bits received (e.g. chase combining or incremental redundancy [123, 119, 29]). Modern PHY layers uses adaptive modulation in combination with HARQ- or ARQ-based error-control coding [64]. 802.11b MAC offers coarse-grained rate-adaptation in response to changing SINR and packet loss conditions. In the context of packetization, the Radio Link Protocol (RLP) [147, 72] used in cellular systems (IS-95, GSM) fragments packets in smaller units (256-500 bits) to limit the impact of bit errors on packets [147, 117]. Hybrid link-layer extensions to RLC also propose applying FEC protection at sub-packet granularity [27].

Berlekamp provides a good summary of ARQ/FEC schemes on the link-layer [29]. In a Type 0 ARQ scheme, protection against losses is offered only through ARQ. No FEC protection is provided. This scheme is applicable to channels which are mostly error free but subject to infrequent interference. In Type I ARQ/FEC schemes, initial transmission is protected using FEC. If this fails, ARQ is used to repeat the transmission. One drawback of this scheme is that FEC redundancy is present even when noise is not. There are several variations of Type II ARQ/FEC. The main characteristic of Type II schemes is that ARQ is used first followed by FEC. Typically, the initial transmission has error detection built in but no error correction. If the initial transmission fails and errors are detected, the sender does not retransmit the original data but sends check bytes which combined with the original transmission can be used to extract the data.

2.2.2 Performance-Enhancing-Proxy (PEP) Schemes

We now study solutions for wireless links (specifically last hop wireless) where enhancements tailored for wireless environments are made to TCP. A good overview of these solutions can be found in [129, 55].

TCP Performance Enhancing Proxies (PEPs) [39] are TCP-aware mechanisms

placed on boundaries where network characteristics change dramatically. PEPs include mechanisms for handling bandwidth asymmetry [45, 25], TCP-aware FEC provisioning [91] or adaptive link frame sizing [48]. 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. While performance enhancing proxies are popular for last-hop wireless links for unencrypted flows, it is inappropriate at every hop of a multi-hop wireless path. Performance enhancing proxies (PEPs) [39] that break TCP end-to-end abstractions also offer limited performance relief.

Balakrishnan et al. provide a good classification of mechanisms designed to improve TCP performance over wireless links [24]. These schemes can be broadly classified into three categories: (a) end-to-end protocols where the TCP sender is aware of the wireless link (b) link-layer protocols which attempt to solve the problem by making the wireless link reliable and (c) split-connection protocols which terminate the TCP connection at the base station and use a special protocol designed for the wireless link. TCP-SACK, ELN, TCP-NewReno [58] are examples of end-to-end protocols. The Snoop protocol [25] is an example of a link-layer protocol with knowledge of higher layer transport protocol. Indirect-TCP[22] and M-TCP [40] are examples of split connection schemes.

2.2.3 Schemes for Multi-hop Scenarios

TCP performance is studied by Fu et al. [61] in a stationary multi-hop 802.11 wireless network using IEEE 802.11b channels. They show that the existence of an optimal TCP window size is tied to the hop count in the multi-hop path. Moreover, since the link-level drop probability is not enough to keep the TCP window tied down to the optimal window size, a link layer scheme called Link RED is proposed to tune the packet dropping probability to stabilize the TCP window size around the optimal value. Adaptive pacing at the link layer is proposed to coordinate channel

access.

ElRakabawy et al. [56] also observe that the ideal TCP window is tied to the hop count. However, instead of changing the link-layer, a TCP mechanism called Adaptive Pacing is proposed that operates by estimating 4-hop propagation delay and the coefficient of variation of recent RTT samples. The authors argue that beyond 4 hops, the interference effects are negligible based on transmitter ranges.

In terms of lossy links, IETF PILC[9] and the 2.5G/3G link-layer committees (3rd Generation Partnership Project (3GPP))[12] have been active in designing link level protocols and making recommendations. Most of the solutions involve making the link layer reliable using hybrid FEC/ARQ [77, 90] and using enhanced SACK options in TCP [104]. Among the characteristics of 2.5G/3G networks, there are spurious timeouts, packet reordering, and packet duplication which all reduce TCP's performance. Current limited solutions include reverting congestion control state in the cases when such events occur [79], adapting the retransmission timer [60] and/or the threshold for duplicate ack [98]. Other mechanisms (with limited impact) discussed include TCP/IP header compression schemes [74], active queue management (AQM) [38], Layer 2 retransmission schemes [40, 22], and packet caching in the case of temporary link outages. Limited Transmit [17] provides a more effective recovery of lost segments when a connection's congestion window is small: it sends a new data segment in response to each of the first two duplicate acknowledgments that arrive at the sender.

The problem of discriminating between congestion losses and packet erasures is yet unsolved [34, 33, 32, 43]. TCP's last line of defense today for Long Fat Networks (LFNs) is TCP-SACK [104], and SACK extension [60] that is still a work-in-progress. SACK extension suggests that when duplicate packets are received, the first block of the SACK option field can be used to report the sequence numbers of the packet that triggered the acknowledgment.

Another class of work includes delay-based approaches and delayed dupacks [143] (imitates the earlier link-layer work Snoop [25]) that manipulates TCP dupacks using delay estimations to determine link characteristics. However, the calculation of appropriate amounts of delay for an arbitrary network topology is unknown.

Biswas et al. propose ExOR [36], an integrated routing and MAC protocol to increase throughput in multi-hop wireless networks. A source wishing to send a packet broadcasts it. All the intermediate nodes that receive the packet run an agreement protocol to decide which one is closest to the destination. This node then broadcasts the packet again and so on. Thus, ExOR chooses each hop of a route after the transmission for that hop, thereby giving each transmission multiple chances to make progress.

Liu et al. present ATCP (Ad hoc TCP) in [92] for mobile *ad hoc* networks where they add a small layer between TCP and IP to improve performance. Their technique is to use feedback from the network to put TCP in one of three states: persist state, congestion control state or retransmit state. The scheme is transparent so that nodes with and without ATCP can setup TCP normally. Moreover, end-to-end semantics are maintained.

2.3 Transport-layer Enhancements

At the transport layer, TCP currently uses ARQ techniques only. Rizzo [120] showed the feasibility of RS coding in software at high speeds. Rizzo[120], Huitema [71] and RFC 2488 [102] suggest using FEC for long delay and lossy wireless links (but no specific protocol is proposed). Proactive/reactive FEC have been proposed for multi-cast erasure channels at the transport layer [125, 41, 114].

Let us consider the history of FEC and its use in transport/link layers. 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 (RS) 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 [41] improve on RS codes for large block sizes in terms of computational complexity, and apply such codes to multicast transport. Nonnenmacher et al [114] 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 (e.g. satellite links), but do not propose a particular solution [120, 71]. RFC 2488 [102] 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 [21] 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%). Baldatoni et al.[85, 99, 100] proposed a version of TCP with FEC (but without adaptivity) that works with small error rates. In particular, Baldantoni [85] 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. Adaptive Forward Error Correction (AFEC) is a scheme [116] where FEC is added to real-time traffic. The objective is to avoid retransmissions since real-time traffic has tight delay and latency constraints. The authors concentrate on a stochastic and control-theoretic treatment of the problem for MPEG video traffic. Moreover, the error rates considered are relatively very small. Bestavros et al. [30] propose a scheme called TCP-Boston that aims to be tolerant to fragmentation in ATM networks by adding redundancy on a per-segment basis.

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 inter-arrival times and loss counts) have not been very successful [33, 32].

For LT-TCP, we use ECN [118] 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 [84],

ELN [23], TCP-ELSA [148]). 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 [103] measures output rate over a combination of low-to-medium bandwidth over wired and modestly lossy wireless links and integrates this measure into TCP congestion control (decreases the congestion window 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 Westwood suggests changing TCP's congestion control paradigm through receiver rate measurements. Westwood [103] uses an output rate estimate for congestion control and survives small error rates (under 5%).

TCP Eifel [98] is a scheme that deals on an end-to-end basis with issues such as sudden delay spikes in GPRS (2.5 G) networks using a modified RTO algorithm. This scheme does not deal with heavy erasure conditions. Delay spikes in 2.5G networks which cause spurious timeouts can be combated end-to-end with such timeout deferral techniques. 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.

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 et.al.[26] 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 et. al. [27] study TCP over links employing hybrid ARQ/FEC. They report that ARQ mechanisms (like 802.11 ARQ with exponential back-off and limited persistence [9]) 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 [27]). These studies and the MIT roofnet measurements clearly indicate the limitations of link-level error resilience techniques.

Sinha et al. propose WTCP [132] , a reliable transport protocol designed to operate efficiently over commercial WWAN networks. WTCP is a rate-based protocol (with rate control done at the receiver) and uses only enhancements on an end-to-end basis. WTCP reports performance improvements of 20-200% over protocols such as Snoop-TCP, TCP-Vegas and TCP-NewReno. However, it is not designed to operate over paths with high loss rates.

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).

In the context of 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 et al.[117] 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 advertised MTU [27]. The value of adapting the fragment unit sizes at the link-layer has also been observed recently [48] . Casetti et al [107] 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%).

2.4 Forward Error Correction (FEC) Codes

Forward Error Correction (FEC) is system of error control where the addition of some redundant information enables the receiver to recover the original data (within some bound). Shannon's theorem predicts the existence of error control codes that can be used to achieve the channel capacity at arbitrarily small error rates. The design of error correcting codes that can deliver this has been the holy grail of digital communications for the past several decades. An overview of available FEC codes can be found in [133, 145, 64]. We provide a cursory look at FEC here.

FEC is a popular method complementary to ARQ. FEC involves designing redundant information for a stream of data so that data lost in transit can be recovered at the destination without the need for retransmission [90]. Though FEC is helpful when bandwidth is plentiful, if the number of total packets lost is greater than the number of redundancy packets, then no data is recovered [83]. For example, Reed-Solomon (RS) codes work only for a small value of N, K and q (alphabet size) and have an implementation complexity (decoding and encoding) of $O(K(N-K)log2N)$ packet operations [105]. Reed-Solomon codes do not degrade gracefully, i.e. if more than $(N-K)$ bits are in error then the message cannot be recovered.

Tornado codes introduced in 1998 have implementation complexity of $O(Kln\frac{K}{\delta})$, where δ is the channel loss rate [41]. Tornado codes are instances of a class of codes called Low-Density Parity Check (LDPC) codes. They are sparse graph codes and are rateless, i.e. a large number of encoded packets can be generated from the original source message. Thereupon, the receiver only needs to collect a number

of bits slightly than the size of the source message to decode the original message (in practice 5% more). Tornado codes are proprietary. However, LDPC and Low-Density Generator Matrix (LDGM) codes encoding/decoding have become publicly available [122].

Hybrid ARQ is an exciting development in 3G networks wherein information blocks are encoded for partial error correction at receiver (i.e. FEC) and additional uncorrected codes are retransmitted (i.e. ARQ). Two main types of HARQ have been proposed - Chase Combining and Incremental Redundancy. Chase combining involves sending a number of repetitions of coded data; decoders combine multiple coded packets before decoding. This scheme achieves gain with small buffer size in a receiver [46]. Type II HARQ or Incremental Redundancy offers a nominal FEC protection, but then sends additional parity bits during each retransmission [77]. Incremental Redundancy requires larger size of buffer in a receiver than Chase Combining [46].

Our innovations at the transport-layer include extending HARQ and transport schemes with FEC codes, and designing combinations of FEC with dynamic window congestion control mechanisms and dynamic MSS mechanisms.

Clearly, the message from prior work is that the potential offered by error control building blocks is yet to be fully realized (especially for operation in extreme environments), and that structuring of such blocks inside protocols and across layers matters. Moreover, we find that all of the schemes presented above deal with relatively small packet error rates (less than 5%). Approaches to dealing with extremely lossy scenarios with bursty losses have not been explored. The rest of this thesis deals with developing mechanisms designed to work under such conditions.

CHAPTER 3
Design Overview and Analysis

3.1 Protocol Design Overview

Our link and transport layer schemes share some common features: adaptive granulation, estimation of loss rate statistics (mean, standard deviation), and FEC units organized proactively (PFEC, along with the original transmission) and reactively (RFEC, in response to feedback). In this chapter, we discuss the design of the fundamental components of the scheme. This design is applicable to both the transport and link layers. In chapters 4 and 5, we look at how the common design can be applied to the link and transport layers respectively. FEC is assumed to be computed using a method similar to that used in CD-ROMs, i.e., shortened Reed-Solomon (RS) codes (see Section 2.4). A portion of the inventory of FEC is sent proactively, and the rest stored for reactive transmission. Alternatively, one could avoid storage and use the Fountain codes or Raptor codes [41, 131] to compute erasure correction units on demand.

Figure 3.1 shows the operation of the general scheme over an abstract lossy channel. Our objective is to provide high goodput across this link while maintaining low delay (i.e. few transmission attempts) and while providing low residual loss rate (loss rate seen by higher layers). We make a distinction between *throughput* which is the total amount of data sent on the link including overheads and *goodput* which is the data usable at the higher layer. For example, FEC packets that go unused and packet headers contribute to throughput but not to goodput.

At the transport layer, we set the block of units (along with PFEC) equal to the current window size while at the link layer, we split a single packet into a number of fragments. As seen in the figure, each set of K data units (packets at TCP or fragments at the link layer) is protected with the addition of $N - K$ proactive FEC (PFEC) to create a block of size N. These N units are sent in the initial transmission attempt. If any K units reach the receiver, the original K data units can be reconstructed. If less than K units arrive uncorrupted at the receiver,

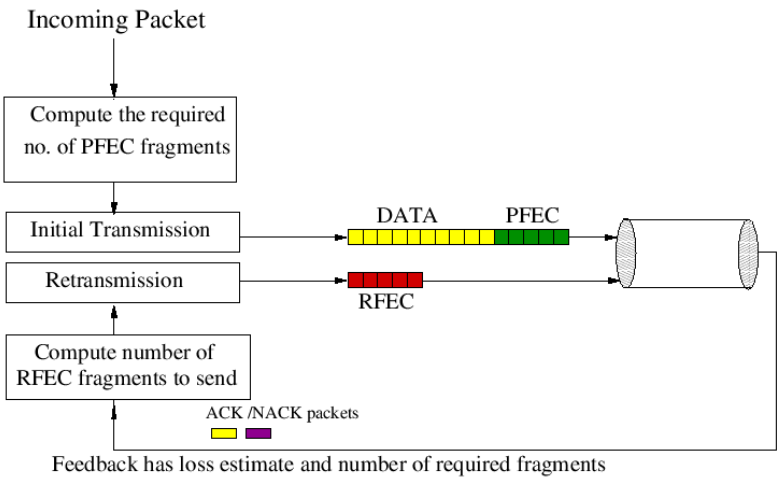


Figure 3.1: Protocol operation over an abstract lossy channel (could be either a single wireless lossy link or an end-to-end path experiencing loss): The initial data+PFEC transmission leads to feedback that determines the amount of RFEC sent to recover that block.

RFEC units are sent to make up for the missing units. Due to the sequence-agnostic property of FEC, the receiver needs to receive any K units (of data or PFEC or RFEC) to reconstruct the original K data units. A question that arises is: how much PFEC and RFEC protection should we provide? How does it affect the trade-off among goodput, latency (weighted average number of ARQ rounds) and residual loss rate? We briefly look at loss rate estimation here and answer the above questions in the remainder of this chapter.

Loss Rate Estimation The amount of Proactive FEC to be provisioned is adaptively tuned based on the current estimate of the loss rate across the channel. This estimate is made based on feedback on the reverse channel (loss estimate is fed back through ack packets). FEC overhead is thus set as a function of the short-term statistics of the loss process. We estimate the loss rate using an exponentially weighted moving average (A) of loss rate samples taken once per block.

$$A = \alpha * sample + (1 - \alpha) * A \tag{3.1}$$

α is chosen differently for the link and transport layers. Note that the EWMA is a fast tracking average for $\alpha >= 0.1$.

3.2 Design Trade-offs and Analytical Modeling

We now develop some analytic guidance for the adaptive portions of the scheme: adaptive granulation, PFEC and RFEC sizing. We model the core HARQ strategy and develop expressions for key metrics such as residual losses, PFEC and RFEC waste. Simulation validation is presented for the model.

3.2.1 Analytical Model

Consider M blocks sent through an ON/OFF erasure channel which has, on average, equal and short ON/OFF periods (shorter compared to the measurement of per-unit loss rate). We refer to this as the Gilbert model for convenience because it resembles a commonly used 2-state Gilbert Markov-model. We also assume that each block is sent either during an ON or an OFF period with equal probability. Each block is divided into N units. We assume that the average erasure rate (PER) p is applied at the granularity of each unit. In our specific example, the ON and OFF periods have an erasure rate of $q = 1.5p$ (with $p \leq 2/3$) and $r = 0.5p$. For p = 50% the ON-OFF period loss rates are 75% and 25% respectively. We therefore capture burstiness as unmeasured bimodality in the underlying loss distribution. This bursty model can be simplified to the uniform per-packet erasure model by setting $q = r = p$. The probability that L, the number of erased units in each block equals i erasures is therefore a bimodal-binomial distribution:

$$P(L = i) = \binom{N}{i} x^i (1 - x)^{(N-i)} \text{ where}$$

$$x = \begin{cases} q & \text{if ON period;} \\ r & \text{if OFF period;} \end{cases}$$

The number of blocks that experience i unit erasures is $M \times P(L = i)$.

3.2.2 Adaptive Granulation and Choice of Block Size N

For a given set of data bytes (either a block of packets at TCP or a packet at the link layer that is comprised of fragments), we resize the units so that we

maintain a minimum level of granulation of the block. At the TCP layer, this policy reduces the probability of losing an entire window and reduces timeouts. At the link layer, it reduces the need for further rounds of ARQ and minimizes the residual loss rate. Fragmentation also allows us to accommodate FEC packets within a block.

Symbol	Meaning
M	No. of blocks
N	No. of units in a block
$P_{fec} = N - K$	No. of PFEC units
X	No. of units needed after first attempt
$R_{fec} = Y$	No. of RFEC units sent
S	No. of RFEC units lost
L	No. of units lost during first attempt
Q	No. of RFEC units that survive
p	Average Packet Error Rate (PER)
q, r	ON-OFF Model State-Loss probabilities

Table 3.1: The various symbols used in the analysis and their meanings are as shown in this table.

In general, we prefer a larger block size (N), i.e. higher granulation to sharply reduce the risk of all units in the block being lost. The reasons are as follows. At the TCP layer, loss of an entire window leads to a timeout, drastic window reduction to a size of 1, and lowering of the *ssthresh* parameter which determines when congestion avoidance begins (and effectively the average window size used). However, a larger block size (in units), while keeping the window size constant (in bytes) implies that each unit is proportionately smaller. Since TCP and IP headers add at least 40 bytes of overhead per unit, the per-packet overheads also increase sharply. The link-layer per-unit overheads are much smaller. Therefore we target a granulation level (N) of 10 units/block for LT-TCP and 20 units/block for the link-layer HARQ (LL-HARQ) scheme.

These choices are explained by considering a binomial distribution model of the number of losses per block (L). This in turn assumes a simple uniform per-unit erasure rate model, i.e. Bernoulli trials for the loss of each unit. For binomial parameters (N, p, q), i.e block size N, per-unit loss rate p and survival rate $q = 1-p$, we are interested in the probability of losing all units. For p = 50%, $N = 5$ gives a

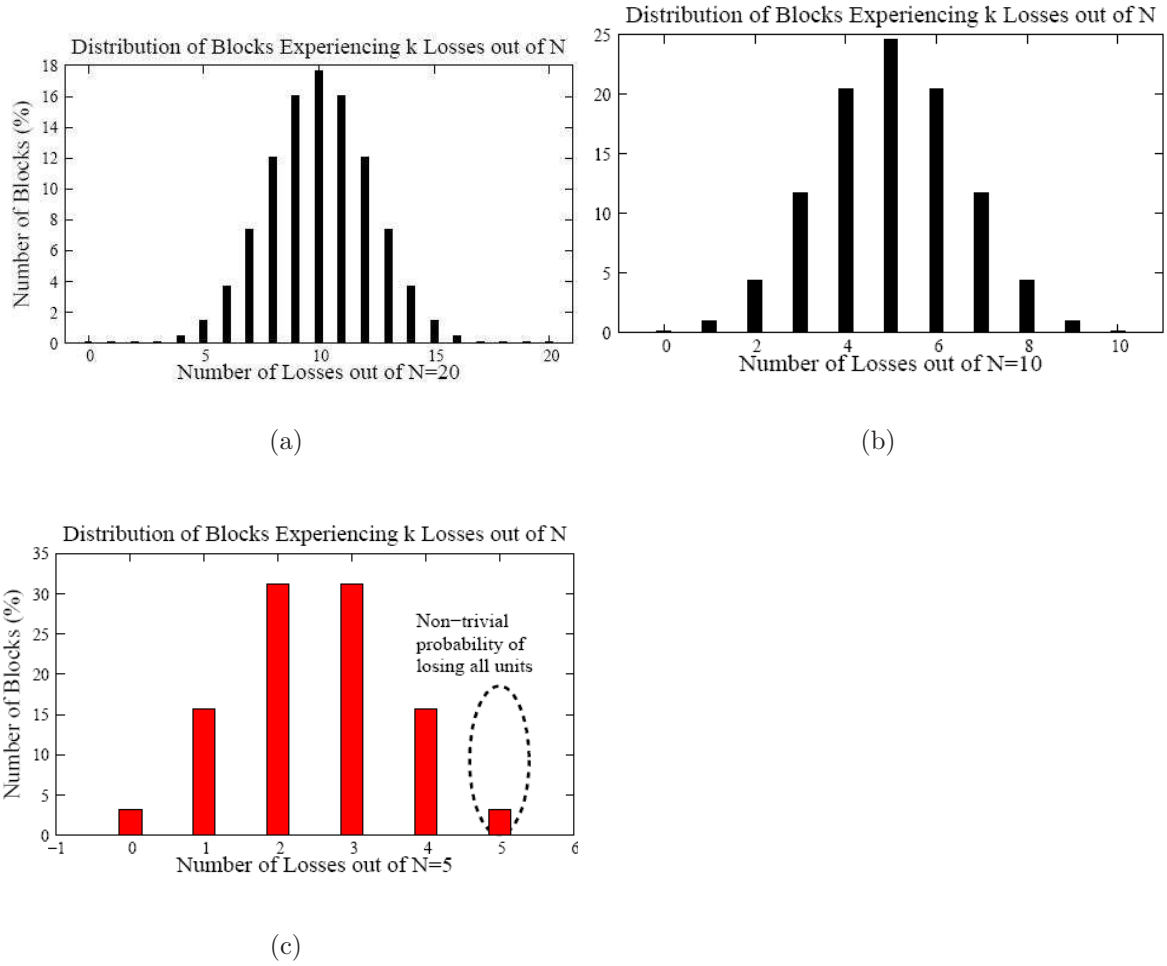


Figure 3.2: These figures show the distribution of losses for three different values of N. This follows a simple binomial distribution. It can be seen that the probability of losing all units is significantly higher smaller the value of N.

non-trivial 3.175% all-unit-loss probability (see Figure 3.2). Increasing N to 10 or 20 dramatically reduces this all-unit-loss probability. If the real unit loss distribution per-block (L) is not binomial (as our scheme assumes), but bursty on small time-scales (i.e. multi-modal), the larger value of N is even more important. For example, if half the blocks experienced a binomial distribution with p = 0.75 and the other half experience p = 0.25 (i.e. a bimodal distribution over all blocks), block sizes $N = 5$, 10 and 20 will lead to all-unit loss probabilities of 12%, 2.8% and 0.1% respectively. So, a block size of at least 10 gives us robustness and timeout risk

reduction even with very bursty underlying loss processes.

A simulation of the bimodal-binomial behavior is shown in Figure 3.3 which plots the frequencies of actual unit losses per block prior to the ARQ attempt (p = 50%, q = 75% and r = 25%); 10 flows, 1-link case. We capture burstiness as unmeasured bimodality in the underlying loss distribution; the two modes (e.g. 75%-25%) are equidistant from the nominal mode (e.g. 50%); and the distance between the two modes is equal to the original mode value.

The larger N is also justified because PFEC and RFEC are provisioned as units within a block. For a given target ratio of FEC in a block, the discrepancy due to round off effects can be significant for $N = 5$, which in turn reduces the achievable goodput. At the same time, we need to consider the per-packet overheads that also reduce goodput for larger N. For TCP, packet sizes of 400 bytes or less will result in header overheads of 10% or more (since TCP+IP headers are 40 bytes at least). Link layers such as HSDPA typically tag individual units with a byte of header. A 400 byte packet when broken into $N = 20$ units leads to a unit size of 20 bytes (160 bits), and a 5% link-layer header penalty (assuming a 1-byte header per-unit). Header compression techniques (if possible end-to-end) would change this trade-off and allow a larger N.

How does this sizing compare to bandwidth-delay products? A 400 byte unit with $N = 10$ implies a per-flow bandwidth-delay product of 4000 bytes or 4 KB. A 10 Mb/s link with 50 ms RTT has a B-D product of 62.5 KB, and can comfortably carry ten such connections (even assuming other overheads). Therefore, our designs are better suited for medium bit-rate links rather than very-small bit-rate bottleneck links.

3.2.3 Adaptive PFEC Sizing

We set the level of PFEC as a function of the mean ($\mu = Np$) and standard deviation ($\sigma = \sqrt{Npq}$) of the assumed binomial distribution of number of losses in a block, where p is the estimated per-packet loss rate. We examine the following choices of PFEC per block: ($\mu, \mu + \sigma$ and $\mu - \sigma$). The reason is that the binomial distribution is well concentrated around the mean (and indeed approximated by

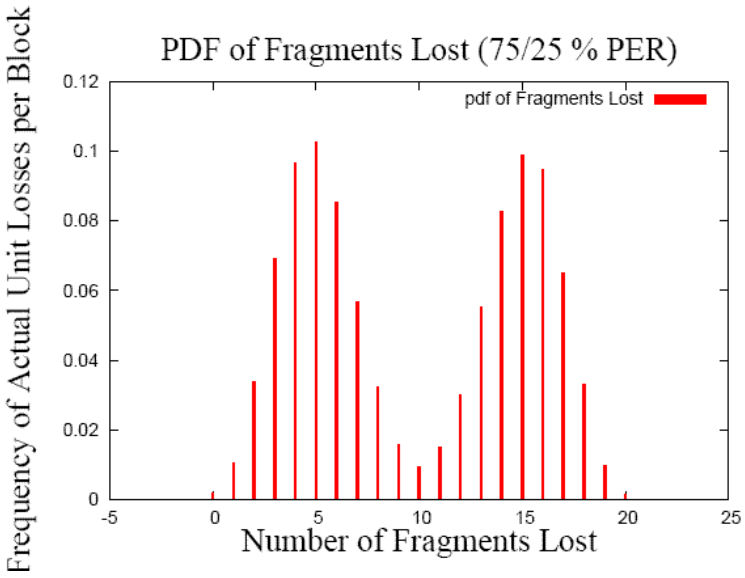


Figure 3.3: Validation of LL-HARQ Analysis. 1-ARQ-attempt Limit: Bimodal Distribution of Lost-Units-Per-Block (prior to ARQ attempt). This reflects ON-OFF PER model with Avg PER = 50%, and binomial modes for 75% ON-PER and 25% OFF-PER.

normal distribution for the special cases when $Npq >> 1$). Setting PFEC either too high or too low relative to the mean would lead to a heavy PFEC wastage or little PFEC impact respectively. This is modeled as follows:

3.2.3.1 PFEC Wastage

PFEC units are wasted (i.e. more than necessary for block decoding) when more than K units arrive, or equivalently, $L < P_fec$. The number of wasted PFEC is $P_fec - L$ in such blocks. The total number of wasted PFEC is calculated as a weighted sum as L goes from zero to P_fec.

$$\#\ \text{PFEC Wasted} = \sum_{i=0}^{P_{fec}} (P_{fec} - i) * P(L = i) * M \tag{3.2}$$

For example, consider a uniform per-packet erasure model and let $N = 20$ and $p = 50\%$; the distribution has a mean (μ) of 10 and a standard deviation (σ) of 2.2 units. The three choices of PFEC above would imply 8, 10 or 12 PFEC units per block respectively (after rounding off). When PFEC = 8 out of 20 units

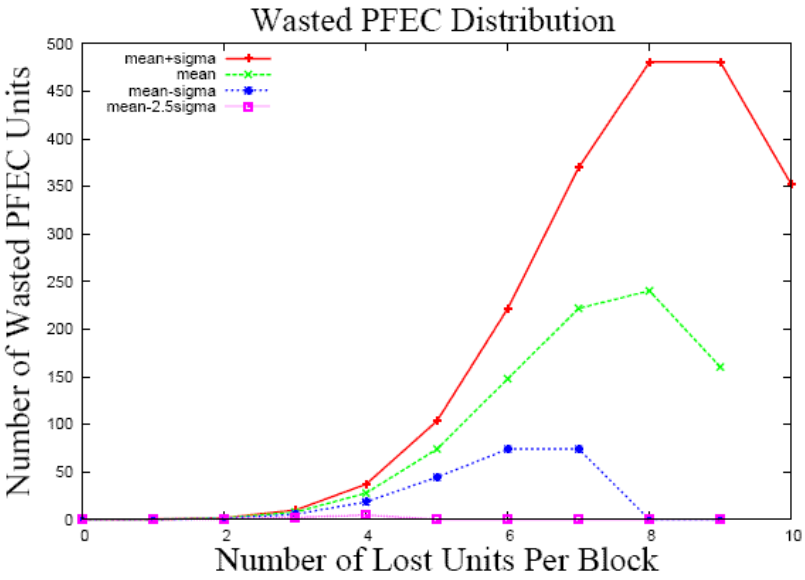


Figure 3.4: PFEC Wastage with different PFEC protection levels: The Y axis shows the number of units wasted with $M = 1000$.

(i.e. $\mu - \sigma$), there is little PFEC wastage (1%), but 75 % of blocks need another round of transmission placing a large burden on RFEC and increasing the timeout risk. When PFEC = 10 units (i.e. μ), there is still low PFEC wastage (4.4%) and 41% need further retransmission. When PFEC = 12 units (i.e. $\mu + \sigma$), the PFEC wastage increases to 11%, but only 13% of blocks need further retransmission and experience timeout risk (see Figure 3.4). This simple analysis suggests a PFEC-per-block choice of $\mu + \sigma$. We accept a higher upfront goodput penalty, to reduce the risk of timeouts (for TCP) or residual loss rate (for the link layer). The analysis also implies that setting PFEC to a very low or very high constant relative to the mean would lead to a poorer tradeoff among expected goodput, residual loss (or timeout risk) and expected latency (number of recovery rounds). In fact, when we set PFEC =25% (i.e. $\mu - 2.5\sigma$in the above case), almost 98% of blocks were un-recovered after the first round!

3.2.4 Adaptive RFEC Sizing

The effect of using PFEC on a binomial distribution is to effectively "chop" the first part of the distribution upto the level of PFEC sizing (see Figure 3.5). The

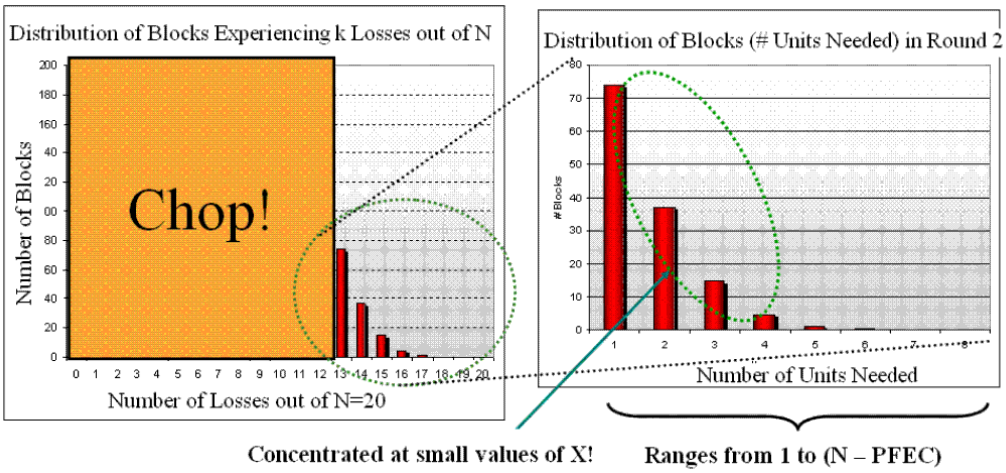


Figure 3.5: Chopped Binomial Distribution of # Units Required for Recovery (X) in Round 2

residual distribution of the units still required for recovery ($X = N - P_{fec} - L$) indicates both the number of blocks that require further recovery and the frequencies for each value of X. This tail decays rapidly ($O(e^{-x^2})$) due to the normal-like tendencies of the original binomial distribution. The distribution also displays a significant concentration at the lowest values of $X = 1, 2, 3$.

Note that we have a dilemma in choosing the value Y not directly captured in the above analysis. If we send RFECs using a simple scaling factor based upon the per-unit loss rate p, i.e. $(Y = \frac{X}{(1-p)})$, the total number of recovery units (Y) sent in this round could be small. As discussed earlier, $Y > 5$, and ideally $Y >= 10$ is desired for minimizing the timeout risk, especially when the underlying loss distribution could be bursty or multi-modal. We propose a conservative provisioning of Y as a function of σ, X and p as depicted in Figure 3.6.

$$\#\text{ RFEC Units sent Y} = \frac{(X + 3\sigma)}{(1 - p)} \tag{3.3}$$

Observe that, the excess RFEC $(Y - X - pY)$, a relative measure of RFEC wastage, could be large. For example, with a Uniform per-packet erasure model, for $X = 1, p = 0.5, N = 20$, Y is provisioned as 15 units and excess RFEC provisioned is 6.5 units out of Y =15 units. In other words, a large number of RFECs could be wasted relative to RFEC sent (43% in this case). However, since the use of

higher PFEC means that a smaller fraction of blocks require RFEC (e.g. 13% of blocks for PFEC $= \mu + \sigma$, when $p = 0.5, N = 20$), and the highest relative RFEC wastage occurs primarily when X is 1, the *total* RFEC wastage is small (3.8%) compared to the *total* PFEC wastage (11%). This total RFEC wastage (assuming a negligible residual loss rate after the ARQ attempt) is a key metric, because it affects goodput. This analysis leads to a non-intuitive insight: **high RFEC over-provisioning can still lead to negligible negative impact on goodput, if PFEC is mildly over-provisioned as well**.

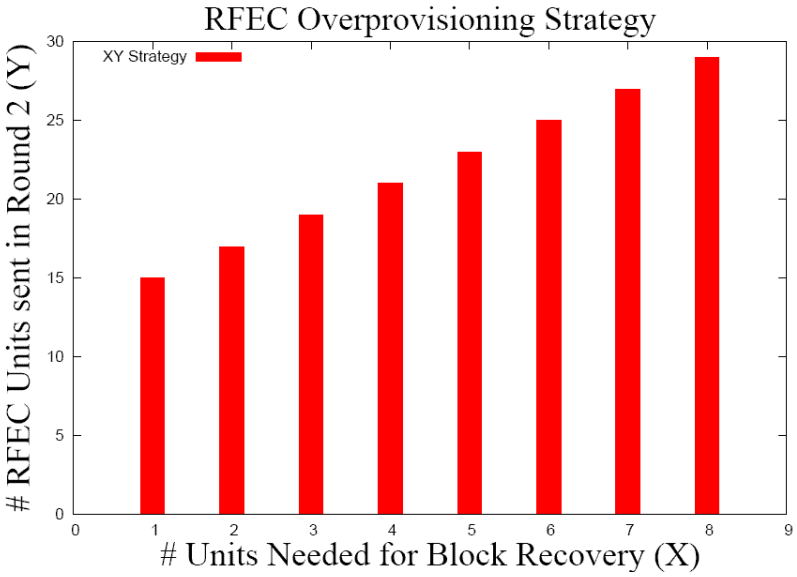


Figure 3.6: The RFEC over-provisioning strategy is as shown. We send a larger relative number of RFEC units form small X to counter the *small N binomial effect*. As X increases, our RFEC response is less aggressive. This strategy in conjunction with slightly over-provisioned PFEC leads to the most favorable performance.

The RFEC over-provisioning policy is also crucial for the bursty/ON-OFF loss process. For example, if the average PER p is 0.5 under the bursty model, our total PFEC and RFEC wastage values grow to 12% and 10% respectively. The higher level of total FEC wastage occurs because of unpredictability and higher-than-expected variance in the underlying distribution. The residual loss rate in this case grows from almost zero to about 3.6%. The residual loss rate is far worse

without the conservative over-provisioning. For example if we use only 1σ instead of 3σ in the formula for Y, the residual loss rate jumps to 12%, even though RFEC wastage reduces to 5%.

3.2.4.1 RFEC Wastage

The total number of RFEC wasted can be computed as a sum over the wasted RFEC for each case when the ARQ is successful. We had sent Y units of RFEC, Q of which arrived, but we needed only X and so $(Q - X)$ represents wasted RFEC.

Now, $[Q - X]^+$ is the positive projection of $(Q - X)$:

$$[Q - X]^+ = \begin{cases} (Q - X) & \text{if } (Q - X) > 0 \\ 0 & \text{if } (Q - X) \leq 0 \end{cases}$$

The total number of wasted RFEC is therefore:

$$\sum_{L=P_{fec}+1}^{N} P(L) * M * P(S = (Y - Q)) * [Q - X]^+ \tag{3.4}$$

The above expression sums up $[Q - X]^+$ with the appropriate probability weights, over the M original blocks.

3.2.5 Residual Losses:

Residual loss occurs when the total number of units after ARQ is still short of K, i.e., $(N - L) + Q < K$, where Q out of Y RFEC units arrive. In this case, we are interested in the probability of residual loss, i.e. residual loss rate. We can rearrange the residual erasure inequality condition above as $Q < (K - N + L)$, or simplified as $Q < X$.

The number of lost RFEC units $S = (Y - Q)$ is distributed bimodal-binomial like the r.v. L seen earlier, but with the conditional variable Y replacing the block size N. Given this, the probability of residual loss $P(Q < X)$ is equal to $P(Y - S) < X)$. This can be rearranged as $P(S > (Y - X))$, which is a CCDF value (complementary CDF value, or tail probability sum) of the bimodal-binomial distribution of r.v. S, the number of lost RFEC units.

The number of residual losses can be computed as the sum over the r.v. $V(L)$, as L ranges from $(N-K)+1$ to N. $V(L)$ is the number of residual losses when L units were lost in the first round prior to the ARQ attempt. $V(L)$ is computed as $V(L) = M * P(L) * P(Q < X)$, where $P(Q < X)$ is the probability of residual loss as explained above.

$$\text{\# Residual Losses} = \sum_{L=P_{fec}+1}^{N} V(L)$$
$$\text{where } V(L) = M * P(L) * P(Q < X) \tag{3.5}$$

The percentages (residual loss rates, % RFEC wasted, % PFEC wasted etc..) can then be computed by multiplying equations(3.2),(3.5),(3.4) with $100/M$. Though this analysis involves several cases and doesn't admit a simple closed form, it is easy to perform numerically for specific values of p.

3.3 Validation and Summary

Error Model	Uniform, (p=0.5)		Uniform, (p=0.1)		Gilbert ON-OFF, (p=0.75/0.25)	
	Analysis	Simulation	Analysis	Simulation	Analysis	Simulation
Link Goodput (Mb/s)	**3.61**	**3.59**	**8.15**	**8.08**	**2.88**	**2.74**
Residual Error Rate (%)	0.0	0.034	0.53	0.0	4.2	4.02
PFEC waste (Mb/s)	1.0	0.99	0.57	0.59	1.13	1.47
RFEC waste (Mb/s)	0.39	0.39	0.28	0.26	0.994	0.662
Avg no. of rounds	1.13	1.12	1.11	1.11	1.90	1.27

Table 3.2: LL-HARQ Comparison of analytic predictions with the validating simulation results [Uniform cases: 10% and 50%; Bursty Case: 75%-25%]. We assume N to be 20 units.

We validated our analysis for two scenarios with the Uniform per-packet erasure model (with p = 0.1, 0.5) and a bursty (Gilbert) error model (with p = 0.5, q = 0.75, r = 0.25). The comparisons between the analysis and simulations can be seen in Table 3.2. The results show that FEC wastage due to over-provisioning is a cause for the loss of goodput, but this is a necessary trade-off to have low residual loss rate after just one round of HARQ. Total PFEC wasted is larger than RFEC

waste despite substantial relative over-provisioning of the latter. Goodput is lower for the bursty case because of inaccuracies in targeting FEC to the loss process. In summary, this analysis points to a trade-off between estimation/prediction errors, FEC-targeting efficiency/goodput, residual loss rate and latency.

This chapter presented the intuition behind the construction of the general protocol design and assembly of the fundamental building blocks. We laid out the trade-offs involved in the addition of these components. We analytically derived the amount of PFEC and RFEC wastage expected along with the expected residual loss rate. We then validated this analysis using simulations for two error models. In subsequent chapters, we apply and tailor the design outlined here to the transport and link layers. We will see that practical considerations force us to deviate slightly from the mechanisms outlined in this chapter (especially at the transport layer). Nevertheless, the analysis performed here yields valuable insights into the nature of trade-offs associated with our scheme.

CHAPTER 4
Design of Loss-Tolerant Transmission Control Protocol (LT-TCP)

Chapter 1 outlined the problems experienced by current variants of TCP such as TCP-SACK. In particular, TCP variants such as TCP-SACK use only retransmissions (ARQ) as their reliability mechanism. In this work, one of the main contributions is the addition of Forward Error Correction (FEC) to TCP to complement ARQ. LT-TCP uses FEC to not only make the transport layer robust against high and bursty loss rates. In this chapter, we outline the design of Loss-Tolerant TCP (LT-TCP) and discuss the reasoning behind the design decisions. The design of the general scheme over an abstract lossy channel was outlined in chapter 3. As discussed, our core design relies on the building blocks of end-to-end loss estimation, FEC provisioned on a proactive (Proactive FEC) as well as a reactive (Reactive FEC) basis and adaptive granulation realized through the segmentation of the TCP data stream. In addition to these, at the transport layer, we also rely on ECN signaling as a means of disambiguating between congestion losses and wireless packet losses. Each of these building blocks is discussed in detail below. While some of the design issues have been posed and answered earlier in chapter 3, we explore these issues in the context of the transport layer and then develop TCP-specific mechanisms to address them. LT-TCP design focuses on the following key issues:

Congestion Response: Since LT-TCP is designed to operate in highly lossy conditions, we need to develop appropriate responses to signals of congestion and packet erasures. We need to design mechanisms so that LT-TCP responds to congestion notifications, but does 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 FEC play in addition to traditional retransmissions (ARQ)? How should the mix be split between proactive and reactive repair?

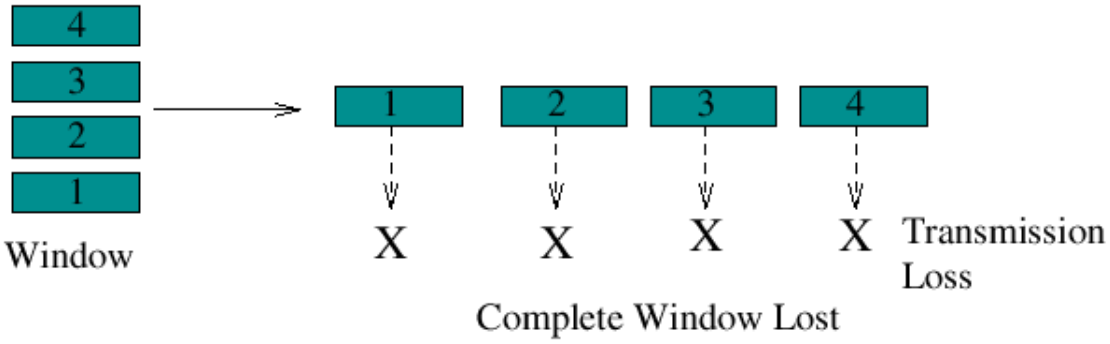


Figure 4.1: This figure shows the incidence of a timeout in the scenario where all the packets in a window are lost. No packets reach the receiver, no ACKs get sent and the sender defaults to a timeout.

Timeout avoidance: Timeouts, though useful as the final fall-back mechanism, are truly wasteful in link utilization. Moreover they cause an increase in application response times. 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. Our approach is also influenced by the fact that the ECN mechanisms have been approved by the standards body (IETF). They have been implemented in all the major operating system networking stacks and have also been deployed on a number of Internet routers. Traditionally, disambiguating between congestion and erasure loss has been considered the primary issue. However, in spite of this simplifying network assumption, there are a number of residual timeout avoidance challenges as discussed below. We show that once this disambiguation is made, the non-trivial task of dealing with loss recovery and congestion control begins. The rest of this chapter is devoted to laying out the design problems related to these two issues and developing enhancements to TCP that can fulfill the promise of adding FEC to the transport layer.

Reliability Mix: As explained earlier, a unique FEC packet can 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 be retransmitted. This sequence-agnostic property of FEC repair packets allows

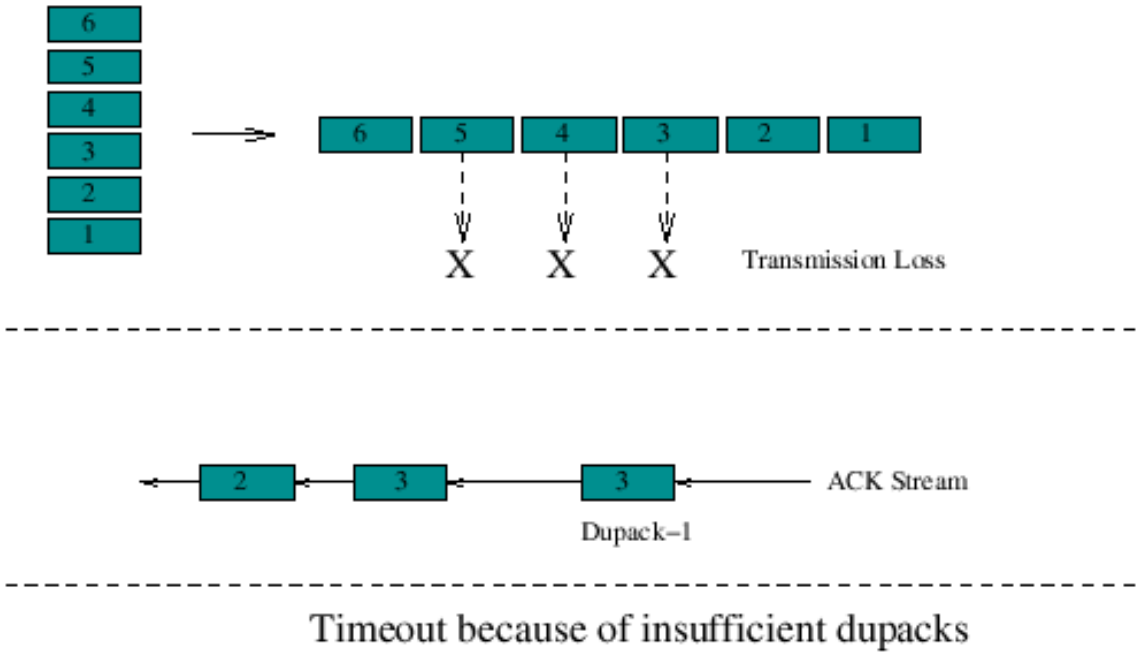


Figure 4.2: This figure shows a timeout that occurs when the number of dupacks reaching the sender is insufficient to trigger a Fast Recovery and Fast Retransmit. Since TCP-SACK reacts on receipt of the third dupack, a small window size (measured in packets) may not be enough to generate the required number of dupacks. This problem can be alleviated by adaptive segmentation so that number of dupacks reaching the sender is increased.

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 fall-back mechanism of traditional repair based on the sequence number of the retransmission or timeout, as discussed below. Our mix of reliability mechanisms 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

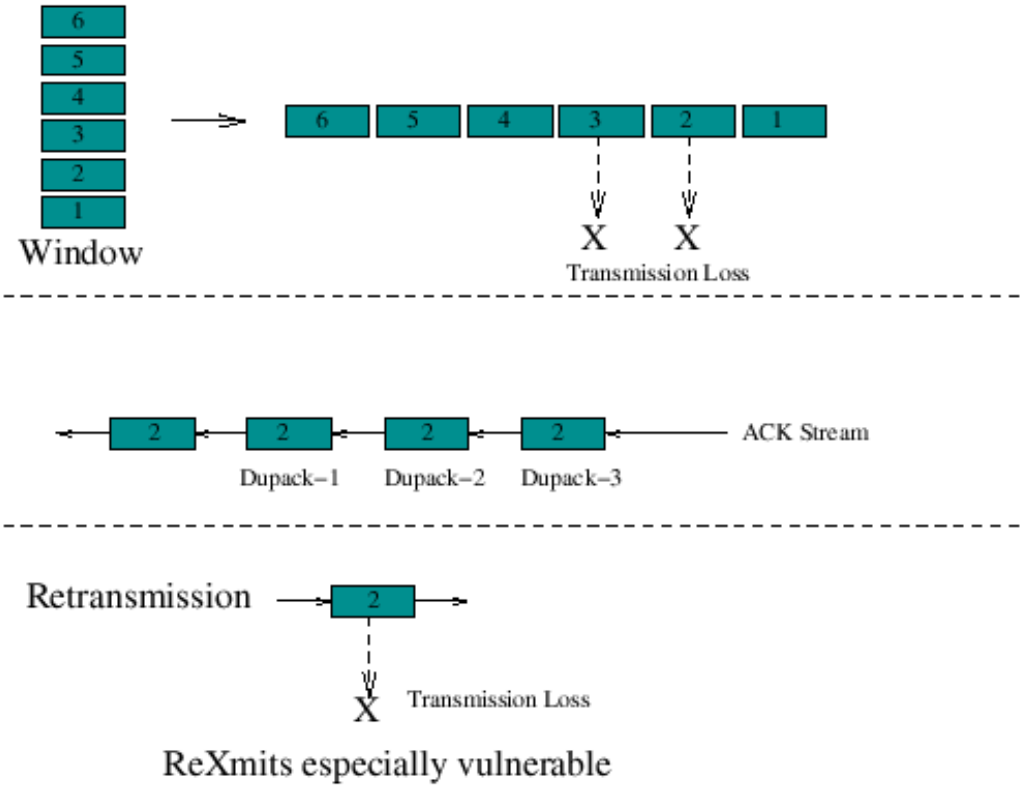


Figure 4.3: This figure shows the timeout resulting when TCP-SACK retransmits a packet but the retransmission is lost. In this scenario, TCP-SACK has no recourse except to fall back to a timeout.

how these reasons are exacerbated in a high packet erasure environment:

1. First, timeouts occur if all packets in a window are lost (see Figure 4.1). 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).

2. Second, even if only a subset of the window is lost, TCP timeouts will if three dupacks do not reach the source (the minimum required to trigger SACK-based retransmissions) (see Figure 4.2). If reverse paths also are subject to link erasures, then timeout probability increases further.

3. 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 (see Figure 4.3). 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 RFEC repair packets triggered by dupacks to complement and protect SACK retransmissions. In the reactive phase, it is better to send RFEC packets instead of retransmissions of data packets since the RFEC packets can stand in for any of the missing data packets. We trigger RFEC transmissions based upon incoming acks (dupacks and 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.

With LT-TCP, we address this issue in three ways:

1. By varying the maximum segment size (MSS) (see Figure 4.6), we allow the window to be more finely granulated (and hence have the capacity to generate dupacks). Granulating the TCP window more finely to increase the number of segments in a window (due to the self-clocking nature of TCP) implies that these segments are spread over an RTT. Smaller packets also reduce the impact of bit errors (which translate to smaller packet error rates).

2. By provisioning PFEC packets in the window based upon an estimate of current erasure rate to reduce the need for dupacks and reduce the burden on the Reactive Phase (i.e. Phase 2). Thus, the current estimate of the loss rate decides the amount of PFEC protection.

3. Use RFEC repair packets triggered by acks/dupacks to recover un-decoded data packets from prior blocks in PHASE 2 transmissions.

In summary, we propose 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.

Per-Window Erasure Rate Estimate: (E) We use an exponential weighted moving average (EWMA) (see 3.1), with $\alpha = 0.5$.

Proactive FEC: FEC packets used in PHASE 1 are called Proactive FEC packets. We add PFEC packets per window as a function of the erasure estimate, i.e., $P = f(E)$. As we will see, if the loss rate is 0, no PFEC packets are provisioned. The overhead is thus scalable and is operative only under lossy conditions. The MSS is adjusted to allow one or more FEC packets per window (while maintaining sufficient window granulation). Finally, computation of FEC is done on a per-window basis using Reed-Solomon (RS) codes. As a side effect, this method yields a large inventory of excess FEC packets for potential RFEC use (see below).

Reactive FEC: Dupacks provide feedback to the sender which is used to send a number of Reactive FEC (RFEC) packets (R) for a given block. R is a function of the erasure rate estimate, E. i.e., $R = g(E)$, number of PFEC packets already sent and number of packets still needed by the receiver to reconstruct the data part of the block. In the Reactive Phase i.e. PHASE 2, we have a choice of the kind of packet to send. For a block that is still un-recovered, we can send either specific data packet retransmissions as specified by TCP-SACK, or only RFEC packets for that block or a mix of both. Clearly, since a FEC packet can stand in for any data packet, it makes sense to send only RFEC repair packets in PHASE 2. While we follow this approach, it should be noted that we can also send data packet retransmissions interspersed with RFEC packets. The advantage of doing that is that data packets reduces the need for decoding at the receiver and will be computationally more efficient. These RFEC packets will complement and protect PHASE 2 transmissions. Observe that there is nothing special about reactive FEC compared to PFEC: 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

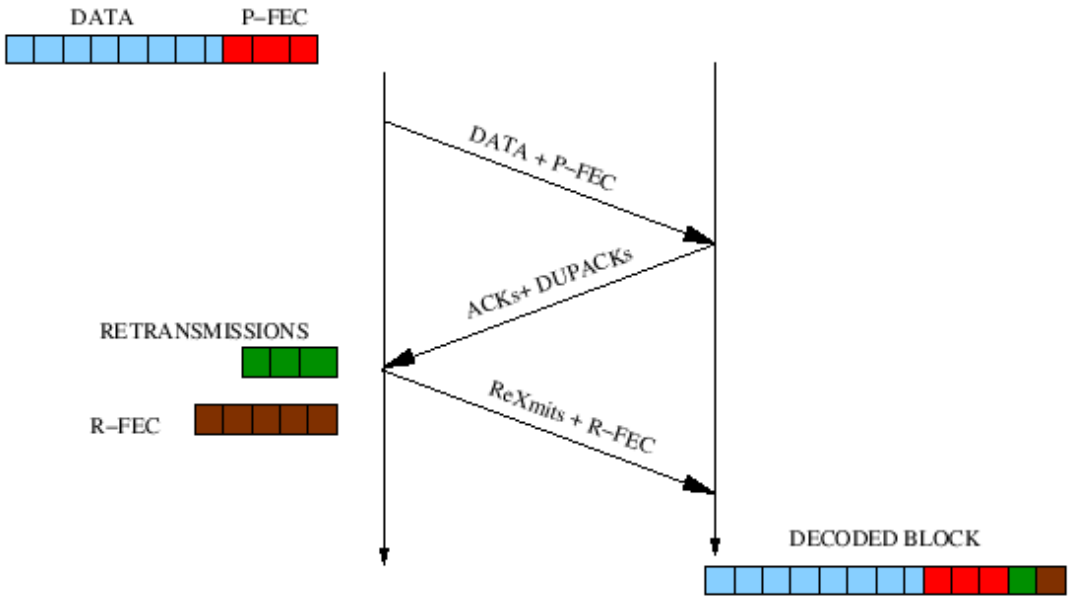


Figure 4.4: Time diagram showing the use of Proactive and Reactive FEC. The initial transmission (Phase 1) consists of data and PFEC segments for the block. Dupacks and acks contain feedback that trigger transmission of RFEC segments and SACK retransmissions.

difficulties in provisioning PFEC packets within the window for a targeted degree of overhead. Without PFEC, there would be an increased reliance on all SACK retransmissions being received correctly and enough dupacks and SACKs reach the source. Without RFEC, 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 trade-offs. 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 during such phases. This overhead diminishes as window sizes grow. PFEC may lead to a small dead-weight goodput degradation due to over-estimation of erasure rate, and some increased burstiness in the release of dupacks from the destination. Reactive FECs triggered by each ack/dupack is provisioned conservatively (i.e. more than strictly needed) and some of it may be wasted as well. 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 to achieve a significant gain in the dynamic range of TCP performance.

In particular, if the loss rate is 0%, then no overhead of FEC (either proactive or reactive) is incurred. However, if the available bandwidth as measured by LT-TCP (measured by *cwnd* i.e. congestion window) is small, it may not be possible to accommodate G MSS_max-sized segments. In this scenario, LT-TCP may incur slightly more packetization overhead compared to TCP-SACK leading a slightly diminished achievable goodput. However, as the available bandwidth increases, this overhead diminishes. Even with medium bandwidth-delay product links, the extra overhead compared to TCP-SACK quickly decreases to 0.

Term	Detail
G	Minimum Window Granularity
MSS_min	Minimum MSS Allowed
MSS_max	Maximum MSS Allowed
new_l	New measured erasure rate
E	Average measured erasure rate
K	Number of data packets in the block
P	Number of PFEC packets in the block
R	Number of RFEC packets in the block
MSS_1	Temporary variable
W	Congestion Window in Packets

Table 4.1: The various symbols used in this chapter and their meanings are as shown in this table.

4.1 LT-TCP Reliability Mechanisms:

We now discuss the LT-TCP enhancements designed to increase reliability and robustness under high loss and bursty error scenarios. While chapter 3 presented the overall framework, the insights offered by the analysis there offers us opportunities to tune the mechanisms to adapt the transport-level mechanisms (see Figure 4.5) as per the requirements at this layer.

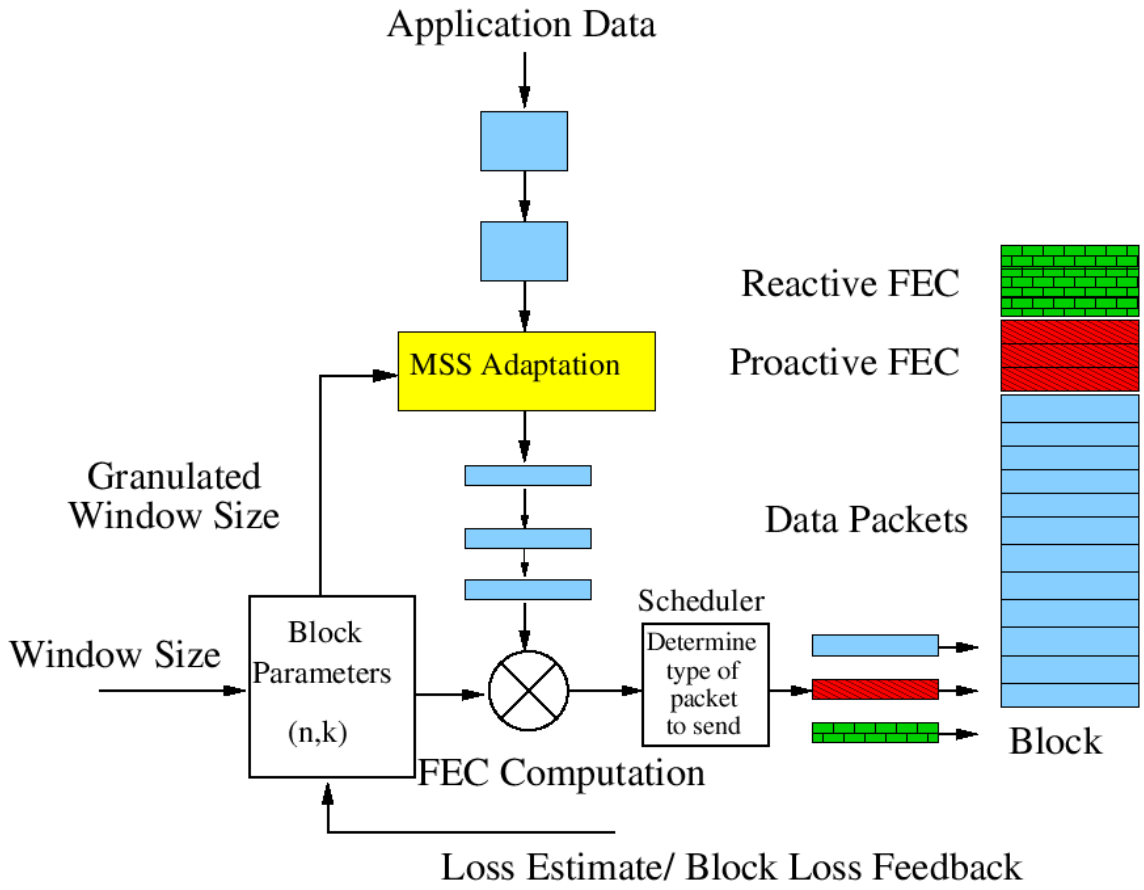


Figure 4.5: Adaptive MSS in combination with PFEC and TCP Congestion Window. The application data is encoded and FEC is generated. Part of the FEC is appended to the data (called PFEC) based on the current loss rate estimate and becomes part of the current window. If this protection is insufficient, FEC is sent in response to number of holes in the block as RFEC. Once any K packets from the block are received, the original data can be reconstructed.

4.1.1 Adaptive Segmentation

As seen in Section 3.2.2, we need at least $N = 5$ to reduce the risk of all the packets getting lost. While at the link layer (chapter 5), we chose a value of 20 for N due to lower header overheads, we make a different choice at the transport layer. For the transport layer, the length of the chunk over which FEC is computed is the current value of the window (e.g., minimum of the congestion and receive window

in TCP). At the transport layer, a window of bytes is divided into equal-sized segments, and FEC is added in units of segments (see Figure 4.5). Segment sizes in different blocks could be different. Like the link layer, we pick the segment size (i.e., MSS parameter) in a block to allow at least one PFEC segment per block if the loss estimate (see below) is non-zero. Furthermore, we have constraints of minimum MSS (200 bytes) and maximum MSS (1500 bytes), and a minimum window granulation requirement ($G = 10$ segments). As the TCP window expands, the MSS is allowed to grow subject to the minimum granulation requirement G. When the TCP window is cut by half (due to ECN-based congestion detection), the window and MSS are recomputed to meet our constraints. The objective is to maintain the window granulation, but not incur unreasonable per-packet overheads due to very small segments. Window granulation serves several purposes in TCP: a lower probability that all segments in the window are lost (reducing timeout risk), ensures sufficient duplicate acknowledgments, and avoids over-provisioning of PFEC (see Figure 4.6).

4.1.2 End-to-End Loss Estimation

For the transport layer, we use per-block loss rate sampling, i.e. the ratio of number of segments lost divided by the total number of segments sent in that block. This per-block loss sample (denoted by *newl*) is fed to equation 3.1 with the α parameter here set to 0.5. This somewhat aggressive averaging still operates on a time-scale of multiple RTTs because it is performed over blocks which in turn are derived from window sizes that get transmitted roughly over the time-scale of the end-to-end RTT. Figure 4.7 shows that Equation 3.1 tracks the average erasure rate fairly well.

The parameter determines the accuracy of the tracking. Having a higher value gives more weight to the current sample and allows us to respond to the changing loss process faster. This is especially true with bursty/Gilbert loss processes where the loss rate may change. For this reason, we opt for a higher value of α even though the accuracy is lower. Under such conditions the PFEC algorithm will add dead-weight 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

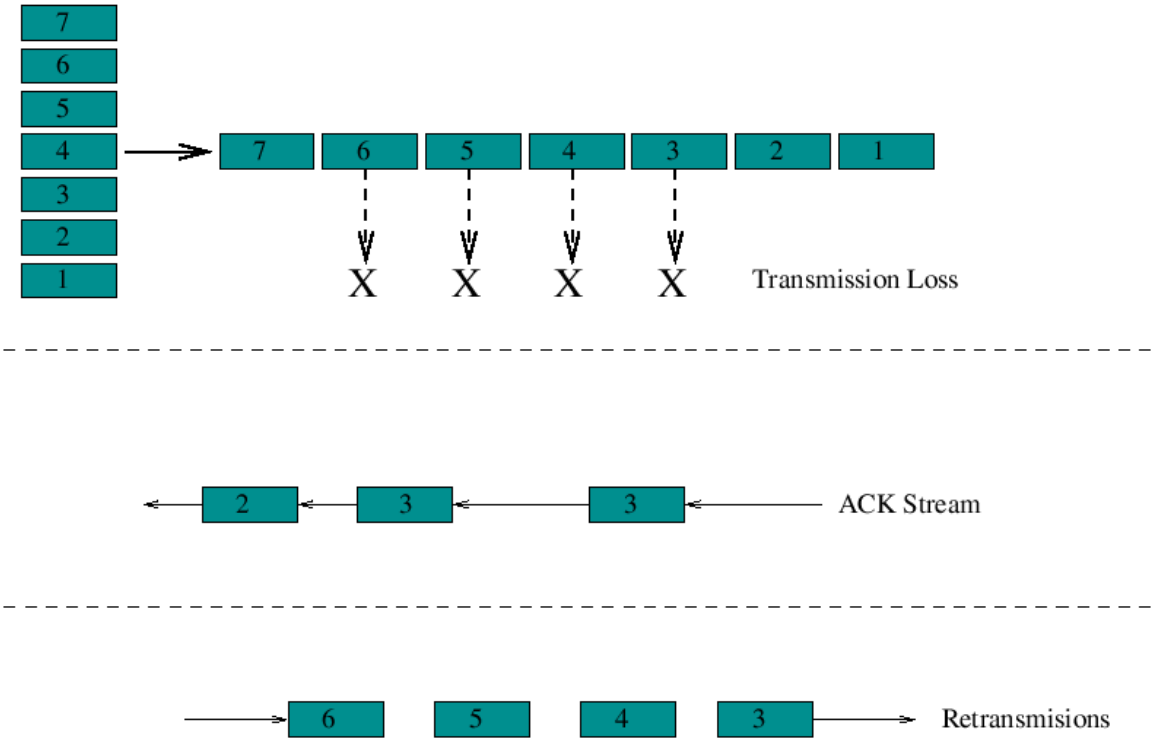
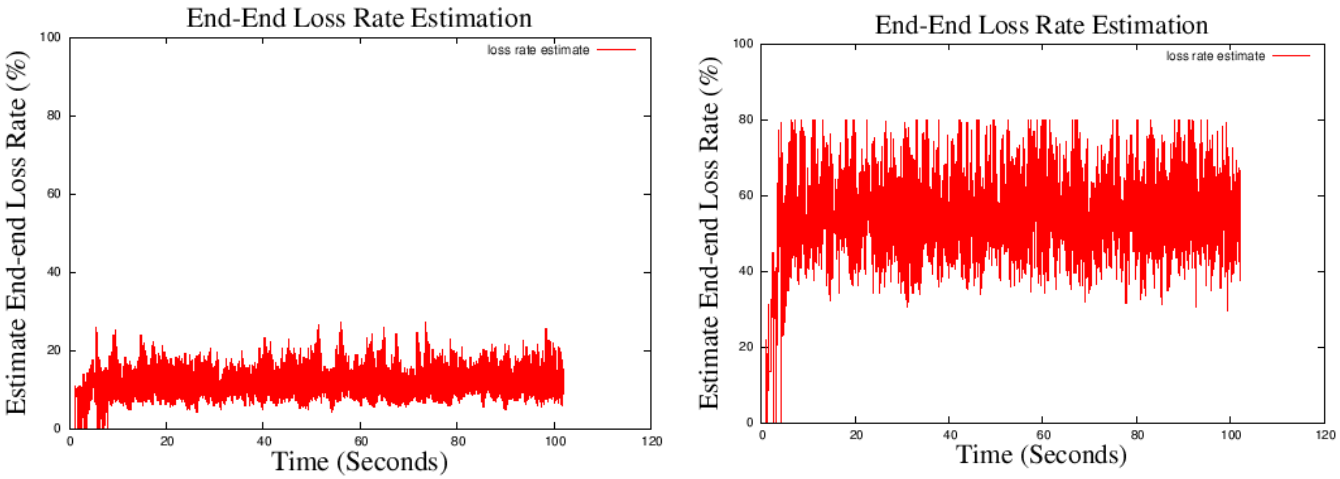


Figure 4.6: The advantage of using adaptive packet sizes can be seen in this figure. Since TCP acks every incoming packet, by splitting the same amount of data bytes into a larger number of packets, more opportunities for generating ACKs are created leading to a persistent flow between the sender and receiver. This helps avoid timeouts.

relatively stable over intervals as large as a second [16], 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.

4.1.3 Proactive FEC Design

As discussed in chapter 3, the PFEC protection should be set to around the mean of the expected loss plus one standard deviation. At the transport layer, the granulation overhead is higher (40 bytes with TCP/IP header). As discussed earlier (see Section 4.1.1), we have chosen a conservative value of 10 segments as the value



(a) End-end Loss Estimation (10%PER)

(b) End-end Loss Estimation (50%PER)

Figure 4.7: End-end loss rate estimate. The underlying loss process is a Uniform loss process with a packet error rate of 10% and 50% respectively. We see that the estimate tracks the underlying loss rate quite accurately.

of N. These are split into K data packets and P_{fec} PFEC packets where the value of P_{fec} is chosen to be the mean of the distribution as discussed in Section 3.2.

The reason for being conservative in setting PFEC at the transport layer is two-fold: (a) we expect the link-layer to reduce the loss rate seen by the transport layer which will lead to lower variance and (b) Transport layer may make multiple retransmissions if needed. Figure 4.8 shows the use of PFEC with a smaller block size (six) compared to the minimum granulation of ($G = 10$) segments for LT-TCP.

4.1.4 Reactive FEC Design

Unlike the link layer (the design of which is discussed in chapter 5), the transport layer needs to use an ack-and-window based clocking mechanism to avoid overloading the network with RFEC. Therefore, we propose to use PFEC (discussed earlier), combined with persistent RFEC as long as acks/dupacks reach the sender. To ensure enough dupacks, we propose that PFEC or RFEC that reach the receiver also generate dupacks. Using SACK-like information, we build up a budget of RFEC segments to send. RFEC packets are sent when acks/dupacks inform the

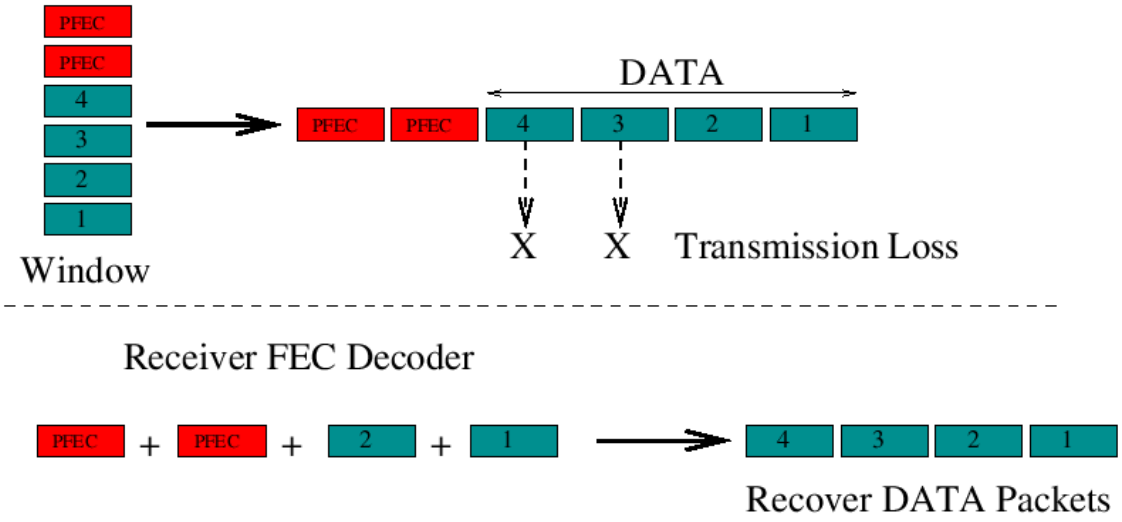


Figure 4.8: This figure shows the use of Proactive FEC (PFEC). Four data packets are sent protected by 2 PFEC packets. The loss of two of the data packets is made up by the receipt of the two PFEC packets. The receiver can use the four received packets to recover the set of data bytes in the four data packets.

sender about segments exiting the network. RFEC packets are scheduled along with pending data and PFEC packets from subsequent blocks while conforming to the principles of self-clocking. This scheduling is explained below. Like TCP Eifel [89], new segment transmission (and RFEC packets) defers the timeout. The use of RFEC is shown in Figure 4.9 for a block size of six.

Once a block has been completely acknowledged (i.e., decoded at the receiver), all pending RFEC and SACK for that block are discarded. After a timeout, we prioritize recovery of outstanding blocks by sending pending RFECs immediately. The structuring of these mechanisms is focused on timeout avoidance (especially avoiding back-to-back timeouts) while avoiding wastage of FEC packets and reduced TCP goodput.

4.1.5 Feedback Design

At the transport layer, the headers of packets from the sender indicate which block the segments belong to. Feedback indicates the currently known cumulative hole size for the last un-recovered block(s) (this can also be inferred from SACK

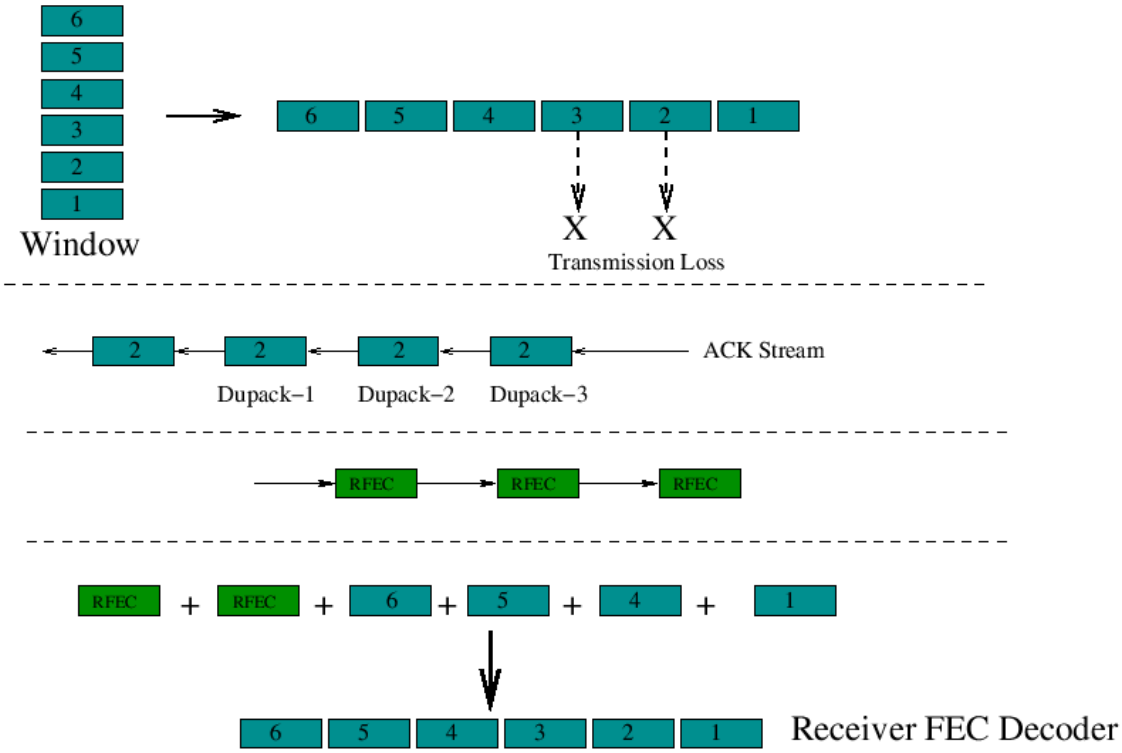


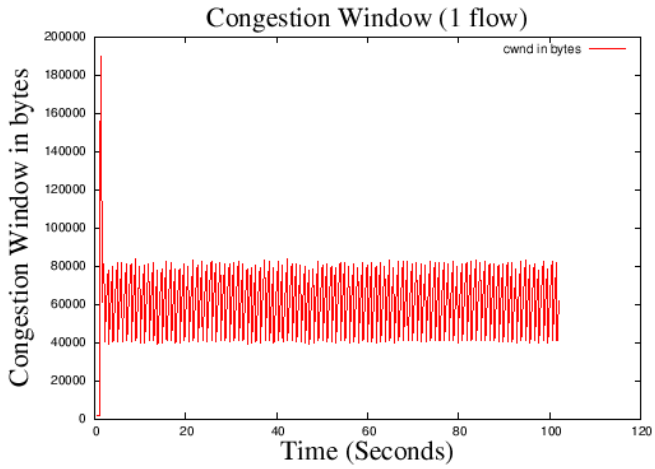
Figure 4.9: This figure shows the use of Reactive FEC (RFEC). In this example, the original set of data bytes has no PFEC protection. The incoming ACK stream at the receiver triggers the transmission if outgoing RFEC packets. The receiver can use the RFEC packets in place of missing data packets to recover the original set of data bytes.

information). Dupacks are generated at the receiver both for out-of-order data and PFEC or RFEC segment reception. Dupacks sent in response to RFEC or SACK retransmissions ensure minimal RFEC persistence beyond one round trip (to further reduce timeout risks).

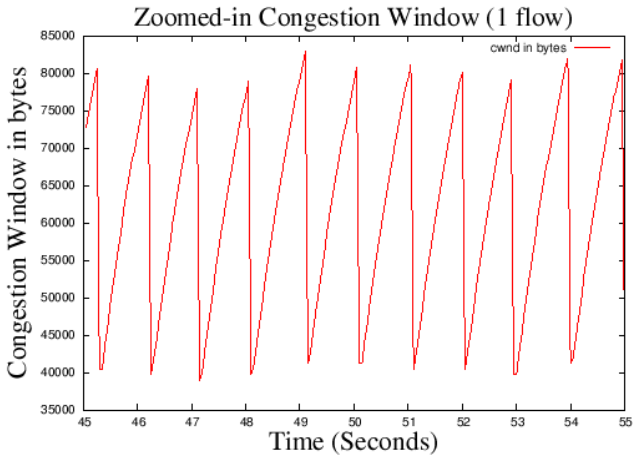
4.1.6 Timer Behavior

The timer behavior in LT-TCP is similar in behavior to that of TCP-SACK. However, since FEC protection is designed to avoid timeouts, we must take care not to timeout prematurely leading to spurious timeouts. This is done by resetting the timer each time we send a packet (data/PFEC/RFEC). Since we expect a steady stream of acks from the receiver, this defers a timeout until we have no feedback for

an extended period of time (RTO interval), implying a genuine timeout. The timer granularity and back-off behavior are exactly the same as in TCP-SACK.



(a) Congestion Window (0%PER)

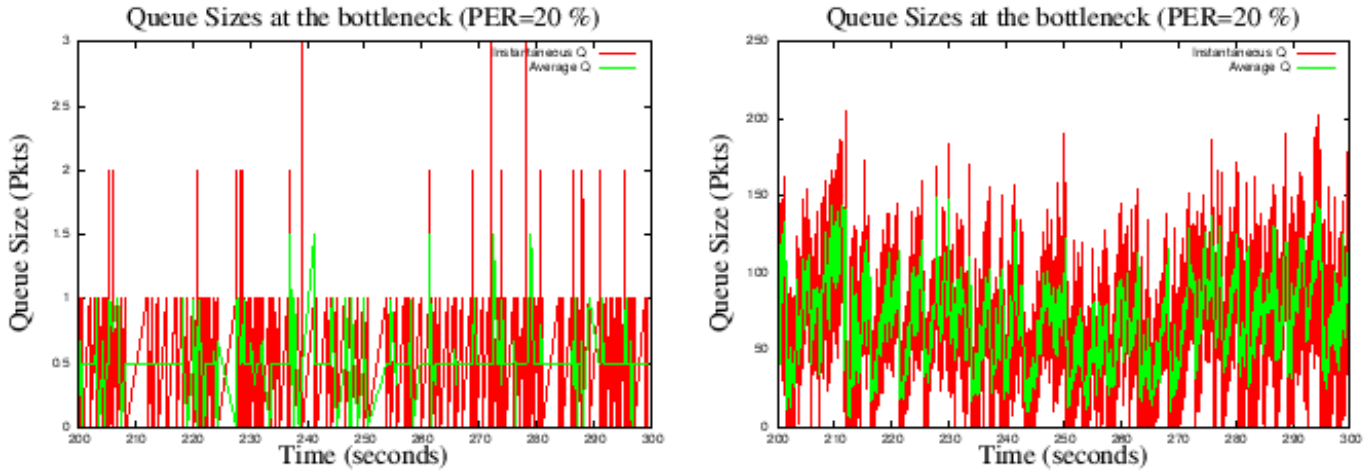


(b) Zoomed-in Congestion Window (0%PER)

Figure 4.10: Congestion window plots for a single flow scenario under 0% PER are shown. The zoomed in plot shows the expected sawtooth behavior.

4.2 LT-TCP Protocol Mechanisms:

The modifications to the TCP sender include a redesign of the TCP transmission engine. While we make modifications to accommodate PFEC and RFEC packets within the window, we still maintain compatibility with the spirit of TCP by adhering to the principles of self-cocking and packet-conservation. These principles dictate that new packets can be sent into the network only if outstanding packets have exited the network. The key changes to the basic TCP sender mechanism include the addition of a scheduling algorithm (see Fig 5.14) that determines which of the following types of packets is to be sent as the window opens up : new data/ PFEC from an upcoming block/ RFEC from a still un-decoded block. A judicious mix of these packets is important to ensure that timeouts are avoided (enough RFEC packets are sent) while wastage is avoided leading to higher goodput (new data/PFEC are sent).



(a) Queuing TCP-SACK (20%PER)

(b) Queuing LT-TCP (20%PER)

Figure 4.11: Queue Length (in packets) vs Time for TCP-SACK and LT-TCP. These simulations were run for a long time (100ms) to remove transient effects.

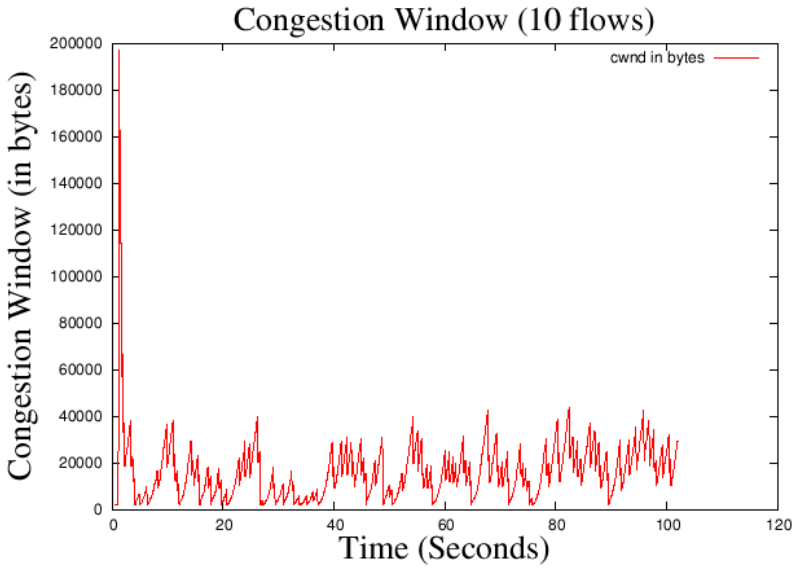


Figure 4.12: Congestion window plot under a 10-flow scenario with 0% PER is shown. The plot is for one of the 10 concurrent flows. The ideal sawtooth behavior is distorted due to the larger number of flows resulting in the behavior as shown.

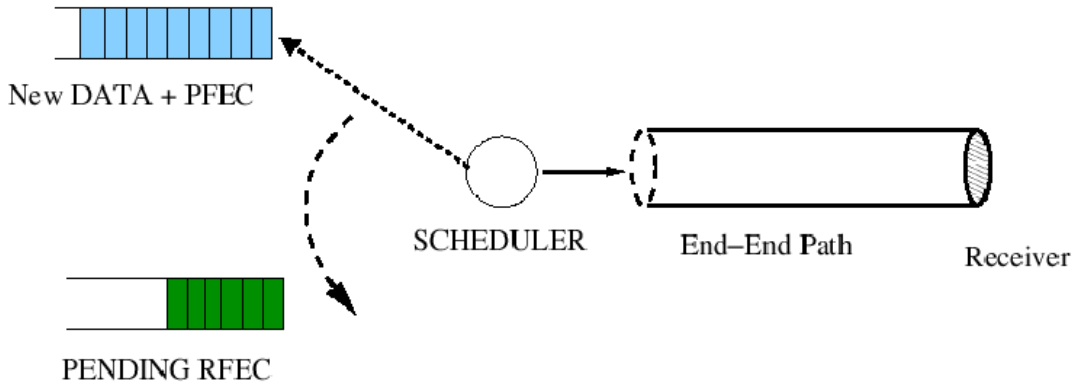


Figure 4.13: This figure shows the scheduling of transport layer packets. Whenever the transport layer protocol gets a chance to send a packet (due to the sliding of the congestion window), it chooses one from two categories. A packet from the current or future block (either data or PFEC) increases obtainable goodput. However, pending RFECs retransmissions are needed to recover data at the receiver and avoid timeouts and are sent first in LT-TCP.

The window growth also needs to be tuned carefully to account for the fact that FEC packets are not part of the sequence space and that acks for FECs *used* FEC packets are acked) ack the highest cumulative decoded data packet so far. To compensate, in addition to normal triggers for congestion window growth, the congestion window is allowed to grow if the incoming ack acknowledges a FEC packet, we are in congestion avoidance and ECN is not signaled in this ack. Moreover, in congestion avoidance, the additive increase component is scaled by $(1 - E)$ in anticipation of the losses expected in flight.

The transmission engine (and hence the scheduling algorithm) is called in response to each incoming ack (and window sliding). In our design, we alternate between new data/PFEC and RFEC packets while maintaining priority for lower-numbered blocks. The amount of RFEC packets to be scheduled for a particular block is computed (RFEC-budget) based on the number of PFEC packets sent, RFEC packets already scheduled, loss estimate and estimate of "hole size" for that block as per equation3.3. By alternating between RFEC packets and new

data/PFEC we obtain a balance between increasing goodput and avoiding timeouts.

One of the criticisms of current TCP stacks is that there a large number of "hacks" that make TCP a complex protocol. Mechanisms such as Fast Recovery/ Fast Retransmit have been present for a number of years. However, recently, several other mechanisms such as Limited Retransmit/ Partial Acks and others have over-complicated TCP. Each of these mechanisms were added in response to perceived problems of losses with the objective of recovering from them. While adherence to TCP semantics is desirable, we are currently investigating if mechanisms such as Fast Recovery, Fast Recovery and Limited Transmit can be removed completely to streamline the sender. Our current solution completely does away with these mechanisms. In a sense, LT-TCP is a much simpler protocol in structure with no hidden and complex interactions. This was made possible by using FEC to solve the problems that we solved using a whole bunch of partial solutions.

Algorithm 1 Calculation of Block Parameters at LT-TCP Sender

1: The end-to-end packet error rate E is estimated continually using the EWMA equation described.
2: Let *cwndbytes* be the congestion window in bytes and *cwnd* be the congestion window in segments. Let *mss* be the current MSS. Clearly, $cwndbytes = cwnd \times mss$. G is the minimum window granulation needed (set to 10).
3: Set mss to MAXMSS (1500 bytes).
4: **while** $N < G$ **do**
5: Compute the number of segments $N = \frac{cwndbytes}{mss}$
6: **if** $N \geq G$ **then**
7: Go to Phase 2.
8: **else**
9: Set $mss = mss - 50$. i.e. reduce the current MSS by 50 bytes.
10: **end if**
11: **end while**
12: Phase 2: We now have N, the number of segments in the window. Each segment is of size mss bytes.
13: Compute the number of PFEC bytes in the window as $pfecbytes = cwndbytes \times p$
14: Compute the number of PFEC segments in the window as No. PFEC Segments is $P = \frac{pfecbytes}{mss}$. Round up to the nearest integer.
15: The number of data segments is set as $K = N - P$

4.2.1 State Maintenance

LT-TCP partitions the data and PFEC byte stream into a series of blocks. Once the size of each block is determined, we populate the K segments with data from the application. We then compute FEC bytes for these data bytes. Part of this is used as the PFEC portion of the block ($N - K$ FEC bytes). The remaining computed FEC bytes are set aside to be used in the RFEC phase. As mentioned earlier in chapter 2, this storage can be avoided by the use of rateless codes such as Digital Fountain codes. The block size will vary as the congestion window size and the current MSS for the block changes. The sender maintains for each outstanding segment (in its send window), the block it belongs to and statistics such as the number of PFEC and RFEC packets sent etc.. This state is used to tailor the number of RFEC segments to send. Note that a lot of this state maintenance such as storing an outstanding segment is already done by TCP-SACK currently.

4.2.2 Sender-side Behavior

The LT-TCP sender behavior is outlined in algorithm 1. The window in bytes is cut only upon receiving ECN signals. The packet size is cut in half if possible or else the number of segments in the window is cut in half. In any event, the window size is cut in half (with the constraint that the minimum window is 2000 bytes). The algorithm shows how the block parameters (block size N, Block MSS , number of PFEC segments P and number of PFEC segments K) are calculated.

4.2.3 Receiver-side Behavior

The LT-TCP receiver performs decoding functions in addition to the regular TCP functions. TCP-SACK already maintains a scoreboard that is used to selectively acknowledge incoming data segments. In addition to this, LT-TCP maintains a state table for the LT-TCP blocks. For each undecoded block whose segments are in the receive window, the receiver stores not only the data segments currently at the receiver (already maintained by TCP-SACK) but also maintains the FEC units that have arrived for the block. Note that there is no need to distinguish between Proactive and Reactive FEC units. While incoming data segments can be acked im-

mediately and sent to the application if it is in order, missing data segments can be reconstructed as soon as K out of N segments from that block arrive. The LT-TCP receiver behavior is outlined in algorithm 2.

Algorithm 2 Receiver-side Behavior of LT-TCP

```
1: for Each new packet do
2:   if data packet then
3:     if data packet is in-sequence then
4:       Send it to application.
5:       Update block statistics in LT-TCP State table.
6:       Ack this data packet.
7:     end if
8:   else if FEC Packet then
9:     if Block already decoded then
10:      Discard this FEC packet without acking it.
11:    else
12:      Update block statistics in LT-TCP State table.
13:      if K out of N packets received for the block then
14:        Decode all the data packets
15:        Deliver them to application layer for processing.
16:      end if
17:      Ack this FEC packet.
18:    end if
19:  end if
20: end for
```

4.3 Preliminary Results and Summary

Figure 4.14 show that LT-TCP does not suffer from the rapid degradation seen with TCP-SACK in Figure 1.3. The drop in performance is more graceful due to its resilience at higher error rates. LT-TCP's reduced sensitivity to RTT and robustness to burstiness enable it to perform better compared to TCP-SACK, especially as the average error rate increases. We look at the performance of these protocols (along with link-level protocols) in detail in chapter 7.

In summary, this chapter presented the design of a robust transport layer protocol called Loss-Tolerant TCP (LT-TCP) designed to operate at *average* packet loss rates as high as 50%. Some of the attractive features of this protocol include adaptive overhead (none when there is no loss), reduction in number of timeouts,

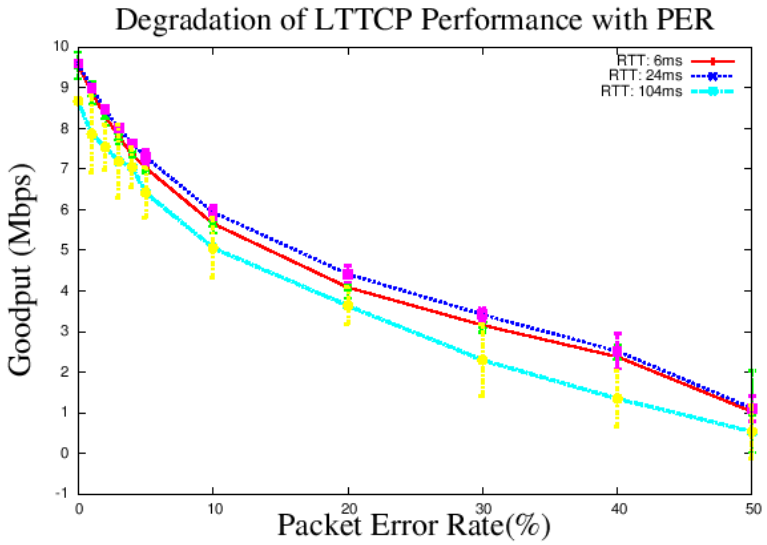


Figure 4.14: LT-TCP performance with Increased Erasure Rate and RTT (Uniform Loss Probabilities, 10 Mb/s Capacity, 1 flow)

adherence to the principles of self-clocking and other TCP semantics and a conceptually simple design backed by insights from analysis. We look at the performance of this protocol under stressful conditions (along with the performance of the LL-HARQ scheme) in chapter 7. We will also study the impact on fairness and latency of LT-TCP. In chapter 5, we will develop a link layer protocol that is designed similarly to LT-TCP. The link protocol is designed to complement LT-TCP while meeting various link-layer objectives. In chapter 6, we see how we can benefit from the use of LT-TCP in a wireless IEEE 802.11 LAN setting.

CHAPTER 5
Design of Link-layer Hybrid FEC-ARQ Protocol (LL-HARQ)

As discussed in chapter 1, the link-layer design objectives include bounded delay, low residual loss rate, high link-goodput, in-order delivery and limited interference with transport mechanisms. Chapter 4 provided a transport-layer scheme (Loss-Tolerant TCP) designed to be robust against high end-to-end and bursty loss rates. An important question is whether a pure link-layer scheme is enough to overcome the performance degradations caused by high loss rates. In this chapter, we investigate this and present a link-layer scheme that attempts to meet our design goals. We then discuss the issue of disruptions and propose enhancements to the basic scheme that can provide good performance even under conditions of disruptions.

In this chapter, we revisit the issue of link-level design with the objective of making it work well under *disruption* channel conditions. In particular, we develop link-layer disruption detection, and respond with a more consecutive mode of link-layer hybrid FEC/ARQ scheme (HARQ) and control the *residual error rate* exported to the transport layer. We define *disruption* as conditions under which a packet (and all its fragments when it is fragmented at the link layer) is lost with certainty. We list below the properties that a link-level design should possess. We develop our mechanisms in the context of a generic link layer that suffers from a high packet error rate including disruptions. Our methods can then be customized and integrated within the context of specific MAC layers. Our link-layer objectives include:

Delay Control: Unbounded number of ARQ-style retransmission attempts on the link-level is not desirable from an end-to-end point of view. Link level mechanisms must be able to provide robustness while keeping the link-level latency small. This limits the number of permissible ARQ retransmission attempts.

Small Residual Loss Rate: *Residual loss/error rate*, defined above should be as small as possible so that upper layers such as TCP are exposed to as small an error rate as possible. This must be done however, while keeping the overhead and

link latency small.

High Link-Level Goodput: Residual loss rate can be driven to zero, either by trading off latency (more ARQ attempts) or by reducing goodput (by adding FEC indiscriminately). The link HARQ scheme must manage this trade-off because the link-level goodput (discounted by residual loss rate) puts an upper bound on end-to-end goodput.

In-order Delivery: The link-layer should attempt to deliver packets *in-order* since delivery of packets out-of-order to protocols such as TCP will cause unintended side-effects (such as fast recovery).

Limited Impact on Transport Layer: The link level mechanisms should interact minimally with the transport layer to avoid effects such as spurious timeouts, variable end-to-end RTT and bandwidth. Ideally, the link should provide TCP with the illusion of a *constant bandwidth, zero-loss* pipe. It is challenging to provide this abstraction in multi-hop, lossy and disruption-prone environments.

5.1 Link Layer Scheme (LL-HARQ)

Our link-layer scheme is based on the design principles outlined in chapter 3. It includes link-layer loss estimation, adaptive fragmentation of a packet into multiple units and FEC provisioned in a proactive (PFEC) and reactive (RFEC) manner. We discuss each of these in detail below.

The protocol is structured to provide a low residual error rate while maintaining low latency and providing high goodput. At the transport layer, reliability is required and hence residual error rate has to be nil. However, at the link layer, we bound the latency tightly by strictly limiting the number of ARQ attempts to 1 (for a total of 2 attempts). Figure 5.1 provides an illustration of the link-level scheme. The loss rate estimate is used to add PFEC to an incoming packet which is then fragmented into 20 units. The initial transmission consists of the data and PFEC units. The receiver sends back information regarding whether additional units are required to recover the packet. Based on the number of units still needed to reach K units, RFEC units are transmitted in the retransmission phase. If the packet cannot be recovered after these 2 transmission attempts, it is dropped.

Link Layer Fragmentation: For the link layer, a single link-layer frame is chosen to be the FEC block, thus avoiding dependencies, and the additional latency, across frames. The link-layer frame is fragmented into units which are subject to potential erasure. The total number of units per frame (N_{link}) is fixed (e.g., 20 units/frame). The size of each unit is finalized once the amount of proactive FEC (PFEC) units per-frame is determined (see below). For our example of 20 units/frame, this policy limits PFEC to be provisioned in multiples of 5%.

Link Layer Loss Rate Estimation: For the link layer, we use per-frame loss rate sampling, i.e., the sample is the ratio of number of units lost divided by the total number of units sent. The sample includes both data and FEC packets with the estimate being obtained separately for the Proactive FEC and Reactive FEC phases. The α parameter in equation 3.1 is set to 0.005 which implies that the averaging effectively spans over 100 frames (or a medium time-scale). A guideline for the time-scale of averaging is that it should be greater than the link-RTT, but be responsive to changes in the longer-term statistics of the loss process. Note that EWMA tends to weigh the recent history of samples using exponentially declining weights dependent upon the parameter .

Link Layer PFEC: For the link layer LL-HARQ protocol, we set N to be 20 fragments based on the discussion above. While a larger N would be better in terms of reducing loss rate, the segmentation overhead will increase. We consider 20 fragments to be a reasonable choice. The current estimate of the loss rate p is used in computing the number of PFEC fragments (P_{fec}) as the mean plus one standard deviation as discussed in Section 3.2. The number of fragments into which the incoming data is divided into is $K = N - P$. These N fragments are sent as the Proactive (PHASE 1) transmission.

Link Layer RFEC: To achieve the goal of limited ARQ persistence, we propose an aggressive RFEC policy: if the receiver at the link layer cannot decode the frame with PFEC (i.e., does not have the requisite K units), we will send RFEC units based on equation 3.1. As seen from the analysis (and shown through simulations), this policy can recover most frames with just 1 retransmission attempt even under highly bursty conditions, without sacrificing goodput. In other words,

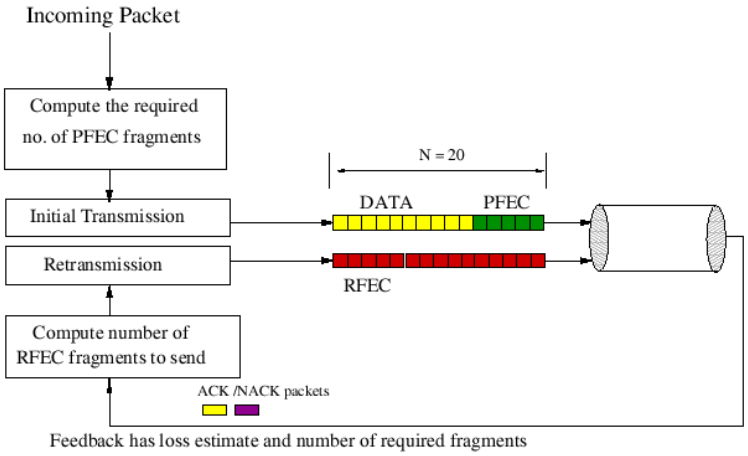


Figure 5.1: Protocol Operation over an abstract lossy channel (could be either a single wireless lossy link or an end-to-end path experiencing loss): The initial data+PFEC transmission leads to feedback that determines the amount of RFEC sent to recover that block.

we do not need a high ARQ persistence to achieve high goodput, if the RFEC is structured appropriately. We argue that despite these modifications, a small residual erasure rate, accompanied by high RTT occurs for the most bursty and persistently lossy cases. Transport level mechanisms are needed to handle these cases, especially as the number of lossy hops increase and the link loss rates aggregate.

Link Layer Feedback Design: At the link-layer, feedback is sent to encode the following information: (a) is the particular frame unrecoverable with just PFEC, and (b) number of fragments still needed to reach the decoding threshold of K units i.e. X. Since units are assumed to arrive in-order in the best case scenario, out-of-order detection of units belonging to the next frame trigger the feedback. Frames are only released in-order at the link-receiver, i.e., after recovery. If insufficient RFEC units arrive and the ARQ persistence limit of 1 is reached, the frame is dropped.

Link-level Flow Control Design: Since links accept units only when capacity is available, and the link-receiver maintains a large buffer (enough to cover the period of ARQ persistence), flow control is simple on the link.

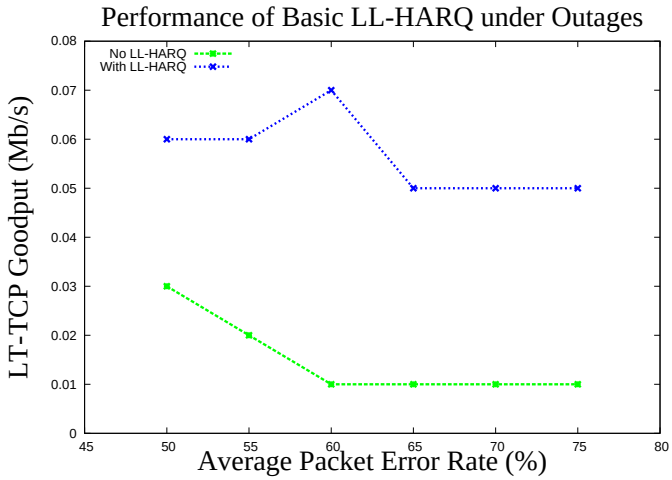


Figure 5.2: The figure shows the performance of the LT-TCP with and without basic LL-HARQ protocol under outage conditions. We see that while LL-HARQ helps in presence of high uniform and bursty losses, it cannot help under outage conditions. We therefore need to add additional mechanisms to help in outage conditions.

5.2 Disruption Enhancements

5.2.1 Need for Additional Mechanisms

While the basic protocol outlined in Section 5.1 provides good performance under uniform and even bursty scenarios (see chapter 7), it suffers under conditions of disruption. Recall that disruption phenomenon is characterized by the periods where the PER is 100%. Figure 5.2 shows the performance of LT-TCP with and without the basic variant of LL-HARQ under such a scenario. The reason for this dismal performance is that periods of disruption lead to burst errors. Loss of consecutive packets on the link lead to timeouts at the transport layer leading to extremely low throughput which is manifested as low throughput/ goodput at the link layer. It is clear that blindly sending data during periods of disruption is counter-productive. The link-layer should behave intelligently and refrain from transmitting during periods of disruption. In the following subsection, we look at augmenting the basic link scheme with disruption detection and response mechanisms.

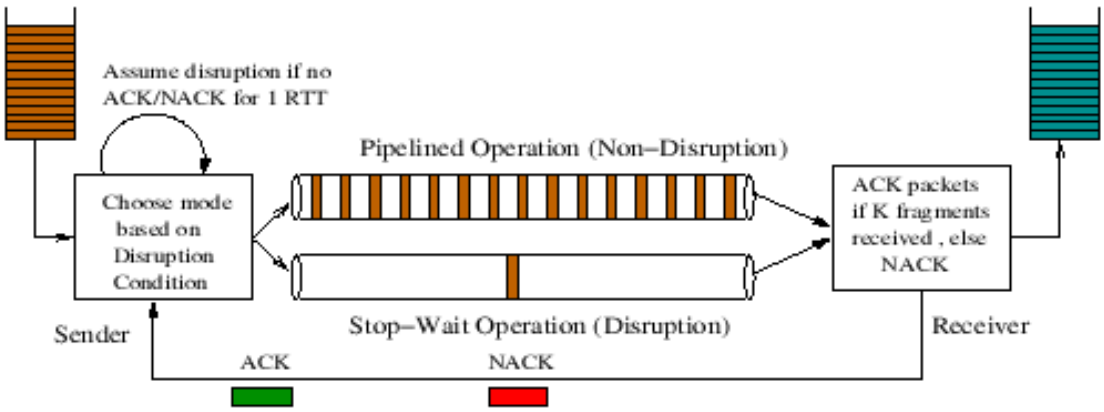


Figure 5.3: Overview of the link level mechanisms: Disruption is detected by the sender and packets are sent in stop-wait mode. Upon exiting disruption mode, packets are sent again in pipelined mode.

5.2.2 Enhancements for Outage Detection and Response

We now look at the specific enhancements to the basic protocol described above that are designed to mitigate disruptions. The key mechanisms are:

Modal Operation: The sender can operate both in stop-wait mode (where each packet has to be acked/nacked before the next packet can be sent) and in pipelined mode (where a window of packets can be sent to fill the pipe). Operating in pipelined mode increases achievable throughput/ goodput but can erase several packets if a disruption event occurs.

Disruption Detection: The receiver is capable of detecting disruption based on a simple heuristic. In our scheme, we assume that if all the fragments of x consecutive packets are corrupted, the channel is in disruption. We set $x = 1$ to detect disruption aggressively.

Retransmission Strategy: A packet can be sent at most *transmissions_plmode* times in pipelined mode and *transmissions_swmode* in stop-wait or probe mode. Once a packet reaches the total number transmission attempts, it is discarded and recovery is left to higher layers. We show in subsequent chapters how a small ARQ limit (both *transmissions_plmode* and *transmissions_swmode* set to 2) is sufficient

Algorithm 3 Basic Link Level Algorithm

1: Receive an incoming packet.
2: Let p be the current estimate of the loss rate on the channel between the sender and receiver.
3: Compute the number of data fragments K such that $K = N(1-p)$ where N is the total number of fragments (set to 20).
4: Compute the size of each fragment = $datasize/K$.
5: Append $N-K$ FEC units to the K units of the same size We now have K data units and $N-K$ FEC units each of size bytes. Note that the total size of the packet has increased with the addition of FEC.
6: A potentially unlimited set of RFEC units is stored with RS coding or dynamically generated if rateless codes are used.
7: Transmit the packet and wait for an ACK or NACK.
8: The feedback indicates the number of fragments still needed to decode the data bytes.
9: **while** ARQ not exceeded or delay-bound not exceeded **do**
10: **if** ACK received **then**
11: Packet has been decoded correctly.
12: Discard the RFEC units for this packet.
13: **else**
14: Let X be the number of packets still needed.
15: Compute the number of RFEC units to send in this retransmission phase based on equation 3.3. Let this be Y .
16: Send the retransmission consisting of Y units.
17: **end if**
18: **end while**

Algorithm 4 Link Level Enhancements for Disruption

1: Start in pipelined mode.
2: **while** outage not detected **do**
3: Run Outage Detection.
4: **if** Outage detected **then**
5: Switch to stop-wait mode
6: **else**
7: Switch to pipelined mode.
8: **end if**
9: **end while**

to yield good performance.

Figure 5.3 illustrates the operation of the disruption detection and response mechanisms and shows the dual mode of operation. The sender chooses the mode of operation based on its detection of disruption. Feedback from the receiver indicates whether the packet was received correctly. If additional units are needed to decode a packet correctly, the feedback indicates this. The figure also shows how the mode of operation is chosen.

In summary, this chapter dealt with tailoring the analysis developed in chapter 3 to the link layer. Based on the insights, we developed a link-layer protocol (LL-HARQ). While the basic protocol works very well under uniform and bursty loss scenarios, it cannot overcome burst errors under outage scenarios. The basic protocol was therefore extended with enhancements so that it could be robust even in the face of disruptions or outages. Due to the simple nature of the link, the protocol can be structured as per the guidelines provided by the analysis. This scheme can be tailored for different MAC protocols such as IEEE 802.11, 802.16 etc.. as per their specific requirements and constraints. The insight here is that by structuring the FEC to cover for the estimated loss rate, we can provide very low residual loss rate while still obtaining high goodput and low latency. Detailed results that show the performance of the LL-HARQ and LT-TCP protocols are presented in chapter 7. The performance under airborne networks is presented in chapter 8 and performance under a wireless LAN setting is presented in chapter 9.

CHAPTER 6
Interference/ Disruption in IEEE 802.11b Systems

The rapid deployment of broadband wireless systems such as 802.11 Wireless LANs (WLANs), 802.16 wireless broadband and neighborhood area wireless networks (see Figures 1.1 and 6.1) raises expectations of high end-to-end performance. In this chapter, we attempt to study the performance of LT-TCP (discussed in chapter 5) over a specific wireless LAN.

We dig deeper into the sources of residual erasures in networks with 802.11based access links or last hop links. In multi-user enterprise/campus LAN environments, a dominant source of erasures is interference (and not channel impairments or noise). Though the purpose of the 802.11 MAC layer is to coordinate multiple user access, it cannot eliminate interference. We therefore consider residual (i.e. loss rate exported to upper layers) interference in the Industrial-Scientific-Medical (ISM) open spectrum bands due to nodes operating with a different technology (e.g., Bluetooth) or due to asymmetric co-channel interference in WiFi leading to capture effects (e.g., hidden nodes in WiFi networks). Our focus is on the interaction between mechanisms at the 802.11 MAC layer and the transport layer in response to such interference-induced packet corruption. In particular, we ask: "*Can* MAC *and transport protocols effectively deliver a significant proportion of the raw bit-rate available at the physical layer to the application in a multi-user environment prone to interference effects?*"

We show that transport-level erasure mitigation opportunities can be significantly limited by PHY level mechanisms (like slow-rate preamble), and aggressive MAC layer mechanisms such as rate-adaptation and persistent ARQ. These mechanisms were originally designed with channel impairments and noise in mind. Since interference is indistinguishable from noise, these mechanisms tend to reduce performance and severely limit other approaches, such as transport layer mitigation mechanisms. In particular, rate-adaptation mechanisms tend to overreact leading to poor channel sharing and increased vulnerability to interference. Some of the

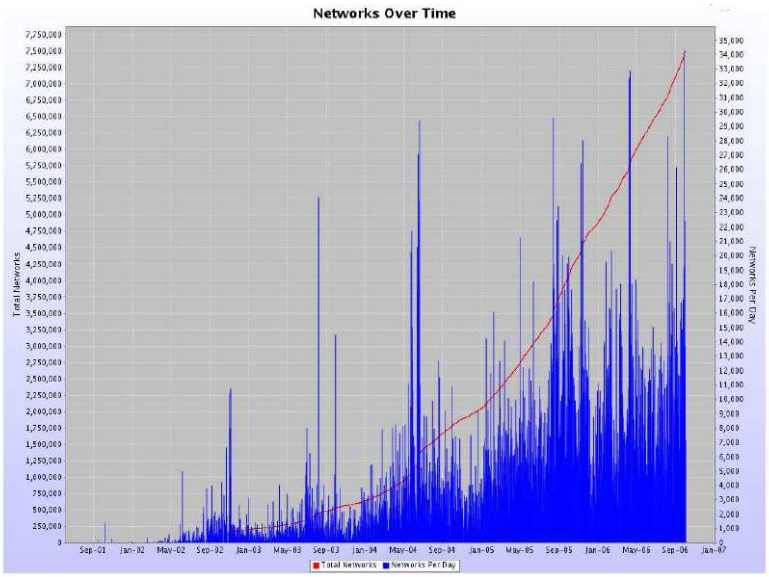


Figure 6.1: This figure (from WiGLE) shows wireless deployment growth for the past several months. The rapid increase in the number of wireless networks deployed clearly poses challenges in terms of achieving good performance in the face of interference and co-channel interference.

current work in rate adaptive techniques is covered in chapter 2.

In the years to come, we envision neighborhood areas that will be serviced by multiple WLAN systems. These coverage areas or cells will be small so that the link quality and capacity is high (from a noise perspective). Moreover, to provide good performance (despite attenuation from walls, floors etc..), cells are designed for the worst case and may be as small as $30m$ radius. However, in typical environments this means that client nodes can associate with multiple APs on the same frequency. This causes a high incidence of hidden node problems with increased adoption and usage of WiFi. In the worst case, it causes severe capture effects. The impact of capture effects can be mitigated by the use of larger buffers (and ECN thresholds) to absorb the burstiness during capture and use of ARQ persistence (drop fewer packets because ARQ attempts to transmit each packet multiple times).

The RTS/CTS mechanisms which were designed to mitigate the hidden node problem are rarely turned on in practice. The reason for this lies in the fact that the overhead incurred is high for the amount of data sent (RTS/CTS are sent at

1 Mb/s). These factors together contribute to a potentially large raw packet error rate. To compensate, WiFi LANs set the number of MAC-level ARQ retransmission attempts to 7 when RTS/CTS is turned off.

We investigate the performance of 802.11 MAC and TCP performance in this context where RTS/CTS capability is turned off. Link-layer ARQ is known to be helpful in WiFi LANs: higher ARQ persistence does decrease residual loss rates and increase resilience to capture effects. The link-level transmission (assuming rate-adaptation) and propagation times are small enough in LANs to allow multiple retransmission attempts. However, the utility of persistent ARQ is affected negatively due to delays induced by exponential timer back-off between successive ARQ retries. Lower latency demands by emerging applications like VoIP-over-WiFi (a.k.a cell-Fi) limit the number of ARQ retries. Longer ARQ retries also do not help in really long capture periods (e.g., beyond 0.5 s capture) because spurious timeouts occur at the TCP level.

The persistent ARQ process also leads to increased per-packet MAC-level overheads and increased vulnerability of further interference because the preamble of all packets (24 bytes) and the entire MAC-level acks (48 bytes) for every ARQ attempt are sent at 1 Mb/s. Since TCP acks also generate multiple ARQ retries and MAC-level acks, the useful TCP goodput with 1500 byte segments on a fully utilized 11 Mb/s link after subtracting out all these per-packet MAC-level overheads is less than 55% if ARQ is done only once (i.e., no interference). The maximum goodput percentage drops rapidly with reduced segment sizes or increased ARQ persistence.

While LT-TCP still performs better than TCP-SACK under such interference-induced multi-layer interactions, reconfiguration of a few key MAC layer mitigation options leads to dramatically improved end-to-end performance.

Briefly, our observations in this chapter show that it is desirable to:

1. Reconsider aggressive rate-adaptation in 802.11 and de facto rate adaptation for MAC level acknowledgments. Preamble can remain at lower rates for safety even though it adds an overhead of more than 3 Mb/s out of 11 Mb/s.

2. Implement LT-TCP improvements to TCP-SACK at the transport layer and ECN at bottleneck queues.

3. Use larger buffers and set higher ECN-triggering thresholds to survive capture effects.

Although it is too late to change 802.11b/g standards, we hope this analysis will inform the debate in 802.11n/ WiMax, and help WiFi network operators better configure existing equipment in enterprises or hot-spots. Our proposal considers heavy packet erasure rates and multi-layer interactions. We propose revised parameter settings at the MAC layer and a new LT-TCP proposal at the TCP layer. The rest of this chapter is organized as follows. Section 6.1 discusses the models for packet corruption considered in this chapter and presents the simulation environment and results. Section 6.2 presents the summary and conclusions of this performance evaluation.

6.1 IEEE 802.11b Simulation Model

Among all the flavors of the IEEE 802.11, we choose 802.11b DSSS (2.4- 2.475 GHz using 22 MHz bandwidth). The RTS/CTS contention avoidance mechanism is turned off as described earlier. With the RTS/CTS mechanism turned off, the number of attempts per packet will be ShortRetryLimit which has a default value of 7. Thus, MAC level ARQ has a persistence of 7 (i.e. 6 retries). Random exponential back-off is used for each retry.

The IEEE 802.11b supports four data rates: 1, 2, 5.5, and 11 Mb/s and multi-rate operation to combat slow fading. Every packet, ack or MAC level ack (macack) has a preamble of 24 bytes sent at the basic rate 1 Mb/s. The implementation and decision basis to change the rate are usually proprietary though some general heuristics are known [87].

However, the implicit assumption is that lowering the rate will decrease the probability of packet error. This is true if the causes of packet corruption involve link impairments alone. However, if the cause of packet corruption is interference, rate adaptation will not help if the signal strength is high enough. In fact, lowering the rate will expose the packet to higher probability of error since the packet is "in the air" for a longer time. In other words, rate adaptation is effective in dealing with propagation losses and not with interference losses. We demonstrate this effect

in the next section.

The simulations were performed using the ns-2 simulator. Six simulations runs were used to obtain each data point of interest. Confidence intervals are shown where applicable.

6.1.1 Cross-System Interference Model

Among various wireless technologies which may produce cross-system interference for 802.11 systems, we choose Bluetooth. Bluetooth headsets for devices such as cell phones are popular and concurrent Bluetooth and WLAN sessions are likely.

Bluetooth wireless links are short range (0-10 m), medium data rate (1 Mb/s) operating in the 2.4 GHz ISM spectrum [54]. HV1, HV2 and HV3 are three packet formats that are used to transmit 64 Kb/s voice over Synchronous Connection Oriented (SCO) links. Typically, Bluetooth headsets operate in the Class 2 mode which is designed for communication up to 10 m with transmission power of 2.5 mW. If the Bluetooth transmitter is close to the WLAN receiver, it can cause WLAN reception bit errors. We adopt a simple Bluetooth interference model: within the duration of a WLAN packet reception, if a Bluetooth hop falls into the WLAN channel frequency range, the WLAN packet is corrupted.

The effect of Bluetooth is modelled through the probability of WLAN packets being corrupted by Bluetooth transmissions occurring near the WLAN receiver. We now begin to derive the WLAN packet corruption probability with a Bluetooth interference source close by (similar to the approach in Shellhammer [130] and Golmie [65]).

A typical Bluetooth voice call uses a full-duplex 64 Kb/s channel. The probability of a Bluetooth packet being on a WLAN channel is dependent on the Bluetooth frame format used. We assume that pure SCO packets (as opposed to hybrid DV (Data-Voice) packets) are used to carry the packetized voice data. For two-way traffic, the three formats HV1, HV2, and HV3 occupy 12, 6 and 4 out of every 12 slots respectively giving Bluetooth slot utilization factors (λ) of 1.0, 0.5 or 0.3 respectively.

We consider an 802.11b channel which occupies 22 MHz of the 79 MHz Blue-

tooth band. Thus, the probability that a Bluetooth packet hops into a WLAN channel is $\frac{22}{79} = 0.27$. We determine the length of each WLAN transmission and the number of Bluetooth slots that the duration of transmission covers. The transmission time of the WLAN packet can be expressed as: transmission time $= \frac{\text{packet size}}{\text{Data rate}}$. Bluetooth has a dwell time of 625 microseconds. Thus, the minimum number of complete Bluetooth slots that overlap the WLAN transmission is $N = \frac{transmissiontime}{dwelltimeofBluetooth}$. The actual number may be one more than this depending on the relative positioning of Bluetooth slots and the WLAN packets. Conservatively, we assume that it is N. For simplicity, we disregard partially overlapping slots. The probability of packet corruption is then given by $PCR = 1.0 - (1.0 - p)^N$ where $p = 0.27 \times \lambda$ is the probability of collision in a specific Bluetooth slot.

Because of the retransmission mechanism of WLAN, not every raw packet loss is visible to higher layers. What is exposed to higher layers is the residual loss rate (defined in chapter 1). If the link-layer ARQ persistence is $K(K \geq 1)$, then the residual packet error rate is given by $PER = PCR^K$ which is the probability that the packet was corrupted on all K attempts.

6.1.2 Simulation Results: Cross-System Interference

We compare the performance of LT-TCP and TCP-SACK over WLAN with and without Rate Adaptation while affected by Bluetooth interference. Since rate adaptation algorithms used in real systems vary from device to device, we used a simple algorithm wherein the transmission rate is lowered (for example from 11 Mb/s to 5.5 Mb/s) when the sender suffers from successive transmission failures. The rate is increased using a hysteresis-based algorithm. Tables 6.1 and 6.2 show the performance of the transport protocols under these conditions. It is clear that operating at the highest data rate is optimal even in the presence of large error rates since the packet is exposed to interference for a shorter duration. The results show that operating at 11 Mb/s enables us to obtain a MAC-level throughput that is close to the maximum obtainable. At lower data rates, repeated packet losses lead to residual losses that lead to timeouts at the TCP level. This limits the flow of data and performance drops drastically. This effect is more pronounced for HV1

interference.

Term	LT-TCP		TCP-SACK	
PARAMETER	Without RA	With RA	Without RA	With RA
TCP-Goodput (Mb/s)	3.74	0.06	2.32	0.005
95 % CI for TCP-Goodput	[3.59,3.88]	[0.05,0.07]	[2.24,2.41]	[0.0,0.01]
Average Number of Timeouts	0	0	0	12.6
MAC Throughput (Mb/s)	5.22	0.54	3.09	0.01

Table 6.1: Performance with and without Rate Adaptation in the Presence of HV3-encoded Bluetooth Voice Calls.

Term	LT-TCP		TCP-SACK	
PARAMETER	Without RA	With RA	Without RA	With RA
TCP-Goodput (Mb/s)	2.83	0.006	0.40	0.0002
95 % CI for TCP-Goodput	[2.61,3.04]	[0.0002,0.001]	[0.37,0.43]	[0,0.0003]
Average Number of Timeouts	0	5.13	13.6	13.8
MAC Throughput (Mb/s)	5.25	0.08	0.65	0.004

Table 6.2: Performance with and without Rate Adaptation in the Presence of HV1-encoded Bluetooth Voice Calls.

Rate adaptation was designed to counter weak signal strength and provide improved spatial coverage for WLAN networks. However, when the source of error is strong interference which affects all data rates equally, rate adaptation is counterproductive. Since we expect future wireless cells to be compact with good-coverage, we need link-layer mechanisms to be robust against interference and not just propagation errors. Our suggestion is to moderate rate adaptation and let higher layers tackle residual errors. For simplicity, the rest of the chapter assumes that rate adaptation has been turned off.

6.1.3 Co-channel Interference Model

In this section, we assume rate adaptation is turned off and that cells operate at 11 Mb/s. Only the preamble for any MAC transmission is sent at 1 Mb/s. We then examine issues with co-channel interference.

Consider the effect of operating different WiFi cells in close proximity in the same frequency channel. Cells more than one cell-hop away typically reuse the spectrum. As mentioned earlier, due to worst-case design constraints, cells could have radii as low as 30 m. While this design improves SNR when there is no interference, it is detrimental when there is a significant amount of co-channel interference.

The packet corruption due to interference is modeled as follows. While a receiver is receiving a frame, if another transmission occurs in its vicinity and the new transmission's observed signal strength exceeds a threshold at the receiver location, the new transmission corrupts the frame currently being received. Interference from multiple sources can also aggregate. It is enough to corrupt a few bits of a packet to render the whole packet useless. However, at high bit rates (11 Mb/s), even 1500 byte packets are short. Further, MAC overheads increase with the number of packets (independent of the size of the packet). Therefore it is better to send larger packets if the bit rate is high (and rate adaptation is turned off).

We assume the transmission range to be 250 m and the interference range to be 500 m. Note that if nodes are separated more than 250 m, the RTS/CTS mechanism may not be enough to prevent hidden node interference. The actual patterns of corruption depend upon the relative locations of nodes in cells and patterns of traffic from the interferer and whether the interferer sees reciprocal interference. Also the impact of losing TCP packets vs losing TCP acks is different at the transport layer (acks are cumulative; packets need retransmission).

6.1.4 Simulations: Co-Channel Interference (Hidden Node)

We use the scenario shown in Figure 6.2. There are two cells: Cell 1 and Cell 2, served by base station 1 (BS-1) and base-station 2 (BS-2). Node 2 is downloading a file from a server adjacent to base-station 2 (BS-2). This leads to packet transmissions by BS-2 that interfere with BS-1. Assume BS-1 is receiving a large file upload from node 1 and relaying it to a remote server (which could be 5ms, 40 ms or 100 ms away). Therefore, BS-1's receptions suffer from corruption due to interference. Since BS-1's transmission of TCP acks or MAC acks are short, and it only interferes with BS-2's reception of short TCP ACKs or short MAC acks (which can be

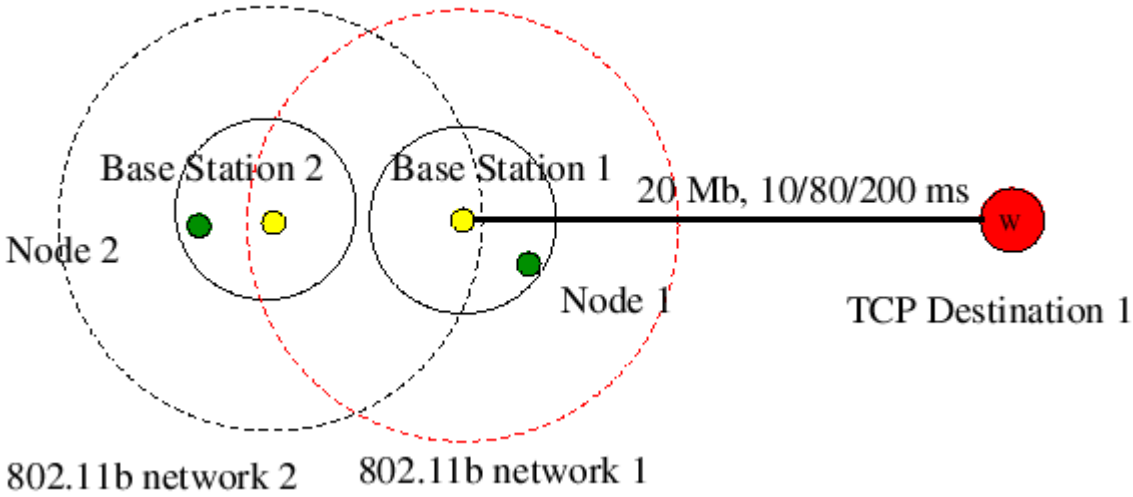


Figure 6.2: Simulation Setup for Co-Channel Hidden Node Interference.

recovered with MAC level ARQ), there is little effect on the download performance seen by node 2. Further, since node 2 sees a short RTT, it ramps up its window faster and essentially "captures" the channel for a period of 250 ms (Node-2's traffic pattern is such that it sends data for 250ms out of every 2 seconds).

Node-1's upload session is effectively shut out for 250 ms every 2 seconds. During this period, each packet at node 1's queue is given to the MAC layer which attempts back-off and retransmission 7 times (roughly 60 ms per packet) before dropping the packet. The TCP layer will see a pattern of no residual loss during periods of no-interference and a huge burst loss during the capture period. In addition, a queue builds up at node 1's IP layer since the MAC layer takes longer to transmit each packet during capture. We therefore recommend careful buffer size settings and conservative RED thresholds to absorb this sudden burstiness and accommodate a larger window to tolerate capture. We will see that LT-TCP's adaptive MSS method will granulate the window to reduce the likelihood that an entire window is lost during capture and that reactive recovery mechanisms work.

Our first set of results (Table 6.3) compare SACK and LT-TCP performance when there is no interference (i.e. Cell 2 is quiet). We vary RTTs to be 10ms, 80ms and 200 ms. These numbers are representative of modes in observed RTT

distributions reported by CAIDA's Skitter measurement project [61]. The short RTT (10 ms) represents intra-metro or intra-regional RTT (e.g., within the Bay area); medium RTTs (80 ms) represents US east-west coast RTTs; and 200ms (and higher) RTTs are observed in transcontinental links (between US, Europe or Asia). The reason we examine multiple RTTs is because even though the WiFi link itself is a LAN link, the end-to-end RTT matters for TCP-SACK when there is even a small residual erasure rate (see Figure 1.3).

Term	LT-TCP			TCP-SACK		
PARAMETER	10ms	80 ms	200ms	10ms	80ms	200ms
TCP-Goodput (Mb/s)	4.43	4.40	4.39	4.64	4.63	4.52
95 % CI for TCP-Goodput	[4.36,4.49]	[4.34,4.46]	[4.34,4.43]	[4.61,4.62]	[4.63,4.65]	[4.45,4.61]
Number of Timeouts	0	0	0	0	0	0
MAC Throughput (Mb/s)	5.70	5.68	5.64	5.89	5.88	5.72

Table 6.3: No interference: LT-TCP and SACK performance without interference under various conditions of end-end delay.

As expected, the goodputs seen by TCP-SACK and LT-TCP are comparable (4.4-4.6 Mb/s) and are close to the maximum possible on 802.11b links with no rate adaptation, and MAC-acks sent at 11 Mb/s regardless of RTT.

In the second set of results (Table 6.4) , we use ARQ = 7 (i.e. six retransmissions at the MAC layer at 11 Mb/s) with 250 ms interference/capture every 2 seconds. Due to exponential back-off, these six retransmissions take upto 60-75ms before a packet is dropped during the capture phase. TCP-SACK goodput improves for both the LAN (10 ms RTT) and USA continental WAN (80 ms RTT) case, although it still collapses for longer RTTs due to high sensitivity to residual error rates. LT-TCP's performance is competitive with TCP-SACK for LANs, and is clearly superior for longer RTTs. This set of results suggests that link level ARQ is not a panacea even with LAN links because the end-to-end RTT still matters. Moreover, such high degrees of ARQ persistence are not possible for longer delay links such as satellite links, which supports the case for end-to-end mechanisms like LT-TCP.

Term	LT-TCP			TCP-SACK		
PARAMETER	10ms	80 ms	200ms	10ms	80ms	200ms
TCP-Goodput (Mb/s)	3.72	3.76	2.54	4.08	3.07	0.37
95 % CI for TCP-Goodput	[3.70,3.74]	[3.69,3.83]	[2.43,2.64]	[4.07,4.09]	[2.98,3.15]	[0.3,0.44]
Number of Timeouts	0	0	0	0	0	25.8
MAC Throughput (Mb/s)	5.24	5.26	3.56	5.44	4.00	0.62

Table 6.4: ARQ = 7, 250 ms / 2 s interference :LT-TCP and TCP-SACK performance with interference of 0.25 seconds out of 2 seconds under conditions of varying end-end delay.

6.2 Summary and Conclusions

The sources of erasures in real wireless networks include both channel impairments (path loss, shadowing, fading) and interference from co-channel and cross-system interfering nodes. For the purpose of understanding their effects on link/transport layers, we make a distinction between uniform erasure losses and erasure losses due to capture effects and interference. In open-spectrum deployments like WiFi, even planned deployments that attempt to maximize wireless coverage by sizing cells conservatively leave scope for co-channel interference.

PHY and MAC layer mechanisms have adaptation techniques designed primarily to handle channel impairments (e.g., rate adaptation, low rate preamble, low-rate control packets like MAC-acks) and export a relatively "clean" virtual link to higher layers. However, these PHY-level adaptive/modulation coding (AMC) or rate adaptation techniques tend to not be appropriate when the primary source of corruption is interference. Such techniques confuse interference as noise (somewhat akin to transport layer mechanisms confusing packet erasure as congestion). Aggressive PHY rate-adaptation response in such situations is counter-productive because the packets are "on-the-air" longer resulting in exacerbating the interference problem. Moreover, it also eliminates possibilities of mitigation at higher layers (link or transport). We demonstrated this effect in the context of Bluetooth interference (i.e., cross-system interference).

We suggest moderation in terms of lower-layer adaptation (especially if there is a significant likelihood of undetected interference), and suggest that hooks be made available for network administrators to turn them off if interference is dominant

in their environments. Link- or transport-layer changes (assuming a constant-rate PHY) work well in this context: larger buffers, flexible AQM parameters for ECN marking, and LT-TCP upgrades for TCP. These enable a large dynamic range of performance (for small and large RTTs, and capture tolerance of at least 250 ms), with a small effect on steady state goodputs.

Capture is a particular form of "disruption" in wireless networks. Disruptions in future networks could occur over longer time-scales, especially in ad-hoc environments. This chapter attempted to solve the problems of interference purely on an end-end basis. In the next chapter, we combine LT-TCP with the LL-HARQ scheme outlined in chapter 5 and evaluate both the schemes in conjunction. We will see that for good performance under really stressful cases (high loss rates with bursty behavior with disruption periods), it is necessary to provide support both at the link and transport layer.

CHAPTER 7
Performance Evaluation

7.1 Simulation Setup

We study and evaluate the performance of the proposed mechanisms in this chapter. We consider a number of scenarios (1-hop, multi-hop) with a wide range of loss rates (0% to 50% PER) under Uniform, Gilbert and Disruption loss models. We also consider the impact of other metrics such as file transfer latency and fairness among flows.

In this chapter, we present the performance of LT-TCP compared with TCP-SACK (with ECN). We then look at the performance of LL-HARQ and compare it with the performance of a baseline link protocol called LL-ARQ. We consider different combinations of the link and transport protocols to study the performance issues. We use both single lossy-bottleneck and multi-hop configurations with 1 and 10 flows with RED/ECN at the queue(s) (see Figures 7.1 and 7.2). Each link is a 10 Mb/s bottleneck link with 20 ms one-way delay) with erasure rates varying from 0% to 50% under different loss models. 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 simulations were run for 100 seconds, and results are averaged over a minimum of 6 randomized runs on the ns-2 simulator [7]. Confidence intervals are shown where applicable.

7.2 Uniform, Gilbert and Outage Error Models

We consider a uniform loss process and a two-state loss process with deterministic error periods (called Gilbert model) to test our scheme (see Figure 7.3). We vary the average PER from 0-50% under both these loss models. We assume that the average or nominal erasure rate (p) is applied at the granularity of each packet. Recall from chapter 3 that the bursty model OFF periods have an erasure rate of $1.5 \times p$ and the ON periods have an erasure rate of $0.5 \times p$. For example, with $p = 50\%$, in the bursty loss model, the link would suffer from loss rates of 25%

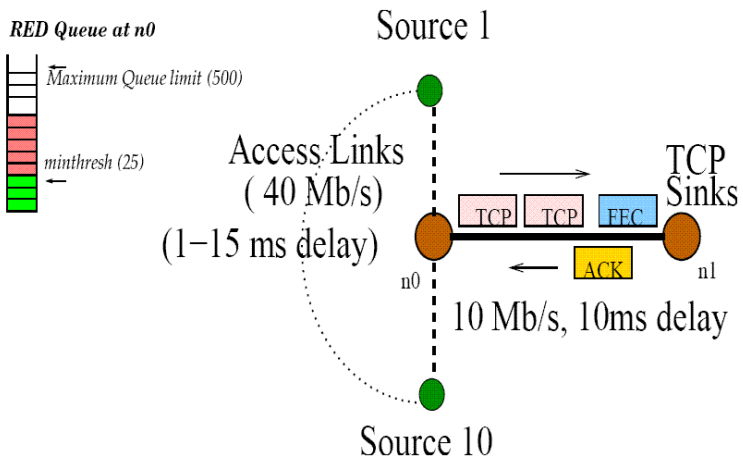


Figure 7.1: Test Configuration: 1 Lossy-link Case

and 75% PER. This can be simplified to the uniform per-packet erasure model by setting the loss rates to be the average PER (p) in both the ON and OFF states. In the example here, in the uniform model, the link error rate in both states would be 50%. Note that while both models have the same average error rate, the bursty model is much more stressful due to variance in the loss rates. In our simulations, the sojourn time in bursty loss model (time spent in either On or OFF states) has a mean of 10 ms, randomized over a small range (9-11ms)(see Figure 7.3). The outage or disruption loss model is similar except that in the ON period, the loss rate is 100% i.e. there is certain loss and in the OFF period the loss rate can go from 0 to 50%. The average loss rate thus goes from 50 to 75% and thus provides an extremely stressful scenario.

7.3 Performance Study

This section evaluates the link (LL-HARQ) and transport protocols (LT-TCP) for a variety of scenarios. This section tests the hypotheses underlying our design and demonstrates the efficacy of our proposed mechanisms with a set of targeted ns-2 based simulations.

LL-ARQ is the baseline link-level protocol which has a pure ARQ mechanism (limit on the number of ARQ attempts set to 10) but without FEC support. To be

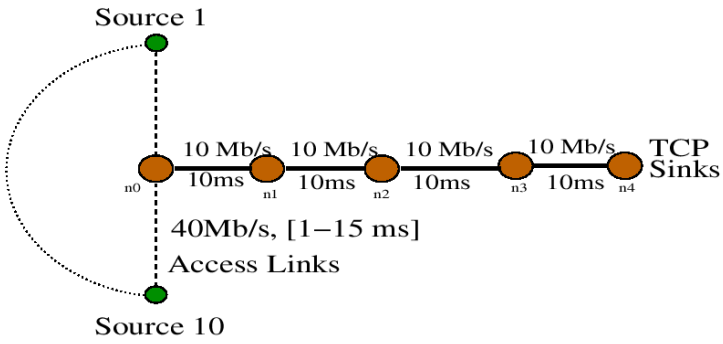


Figure 7.2: Test Configuration: Multi-hop Case

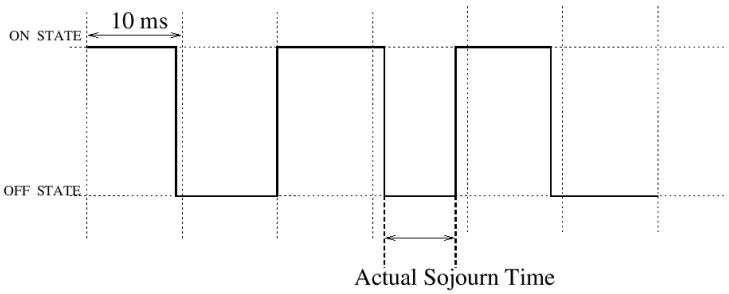


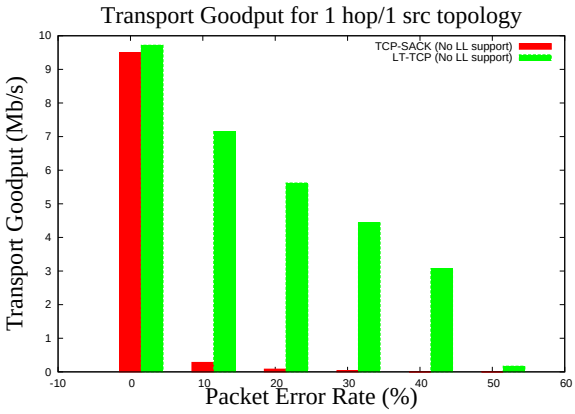
Figure 7.3: The behavior of the bursty error process is as shown. The PER in the OFF state is lower than that in the ON state. For disruption error models, the PER in the ON state is 100% (packets are lost with certainty). For the uniform error process the PER in the 2 states is equal. The actual sojourn time in a state is randomized between 9 and 11 ms.

conservative, LL-ARQ does not incorporate the typical back-off mechanisms between ARQ retransmissions. **LL-HARQ** is our proposed link protocol which has a limit at most one ARQ retransmission attempt and includes PFEC and RFEC mechanisms as explained in section 3. TCP-SACK and LT-TCP are the transport layer protocols used.

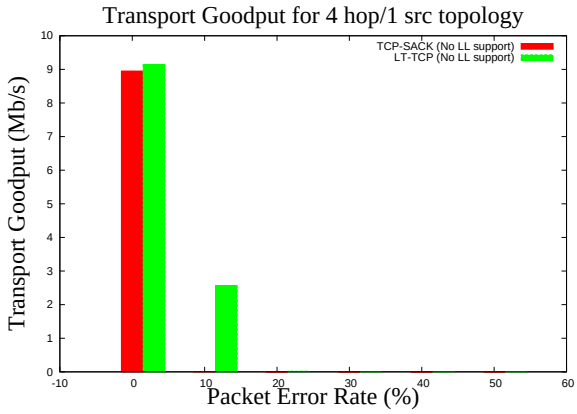
7.3.1 Basic LT-TCP Performance

In this subsection, we look at the performance of LT-TCP in detail with 1 and 10 sources on 1 hop and 4 hop topologies. The link layer is a basic variant with no FEC or ARQ support.

Figures 7.4(a) and 7.4(b) show the performance of LT-TCP and TCP-SACK

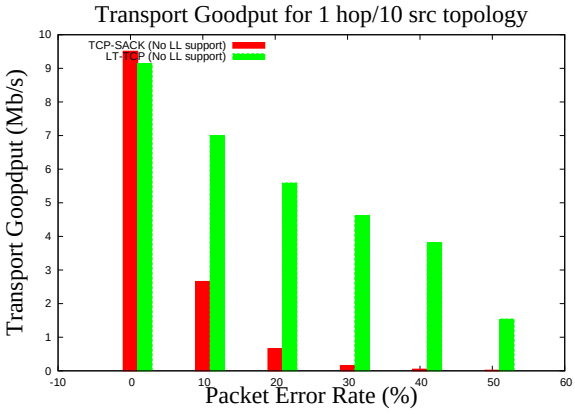


(a) LT-TCP Performance (No Link Support): 1 source, 1 hop

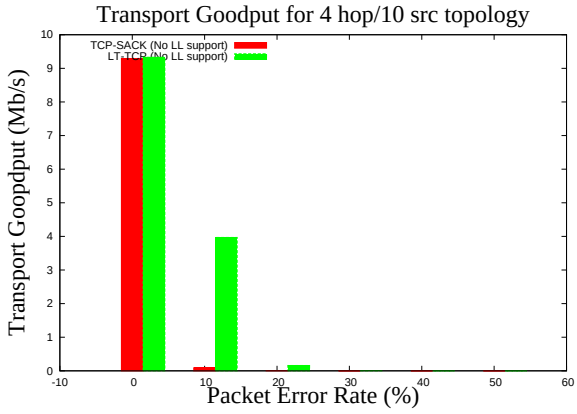


(b) LT-TCP Performance (No Link Support): 1 source, 4 hops

Figure 7.4: This figure shows the TCP-SACK and LT-TCP goodput for the single hop and four hop topology with a single source. The error model used is the Bursty error model. We see that TCP-SACK cannot perform even under relatively small error rates whereas LT-TCP can deliver good performance.



(a) LT-TCP Performance (No Link Support): 10 sources, 1 hop



(b) LT-TCP Performance (No Link Support): 10 sources, 4hops

Figure 7.5: This figure shows the TCP-SACK and LT-TCP goodput for the single hop and four hop topology with 10 sources. The error model used is the Bursty error model. We see that in the case of a single hop, the performance degradation is graceful. However, with 4 hops, LT-TCP is unable to cope without link support.

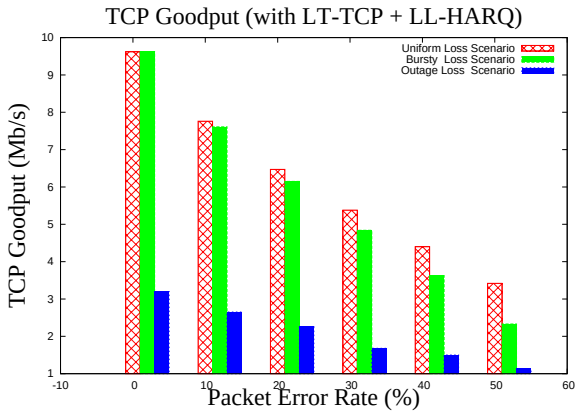


Figure 7.6: The figure shows the performance of LL-HARQ (with LT-TCP) for Uniform, Bursty and Outage scenarios. The X axis plots the nominal PER for the uniform case, the average PER. For the outage case, and the OFF-state error rate for outage scenarios.

without link-layer support (a packet is sent once) over 1 and 4 hop topologies with a single transport source. Figures 7.5(a) and 7.5(b) show the performance with 10 sources. We see that the performance degradation of LT-TCP is much more graceful as compared to TCP-SACK. At the link layer, we have no FEC or ARQ support. We see that while LT-TCP performs better than TCP-SACK, over 4 hops it is unable to cope with the extremely high end-to-end loss rate.

7.3.2 Basic LL-HARQ Performance

In this subsection, we look at the performance of LL-HARQ in detail with 10 LT-TCP transport flows over a 1 hop topology. We look at three different error models (uniform, bursty and disruption). Figure 7.6 shows the performance obtained at the transport layer. Note that the X axis shows the nominal error rate (p) discussed in section 7.2. Table 7.1 shows the detailed performance of the LL-HARQ protocol for the uniform error model. We see that even under the worst case scenario of 50% PER, the average link latency is less than 15 ms. Moreover, the residual loss rate that is exported to the transport layer is nil under all the error rates. Note also that the number of timeouts experienced by the transport layer is very small even under high error rates.

Single Flow, Single-link	ERROR RATE					
PARAMETER	0 %	10 %	20 %	30 %	40 %	50 %
Per-TCP connection Statistics						
Receiver Goodput(Mb/s)	9.62	7.76	6.47	5.38	4.40	3.42
CI Bounds (Goodput)	[9.58,9.66]	[7.72,7.81]	[6.44,6.51]	[5.34,5.41]	[4.38,4.43]	[3.39,3.46]
TCP Timeouts	0.00	0.00	0.03	0.02	0.03	0.17
Sender Throughput (Mb/s)	9.63	7.78	6.49	5.39	4.42	3.44
Receiver Throughput (Mb/s)	9.96	8.05	6.71	5.60	4.58	3.59
Per-link Statistics						
Link Level Goodput (Mb/s)	9.96	8.05	6.71	5.60	4.58	3.59
Residual Loss Rate (%)	0.00	0.00	0.00	0.00	0.00	0.00
Average Link Latency	10.98	13.82	13.25	13.89	14.60	14.98
Link Level Throughput	9.99	9.98	9.99	9.99	9.99	9.99
Link Level Goodput	9.96	8.05	6.71	5.60	4.58	3.59
Link Level PFEC Sent	0.02	1.48	2.90	3.77	4.60	5.40
Link Level RFEC Sent	0.00	0.38	0.36	0.59	0.80	0.98
Link Level PFEC Wasted	0.02	0.61	1.04	1.04	1.02	1.00
Link Level RFEC Wasted	0.00	0.26	0.22	0.33	0.38	0.39

Table 7.1: The table shows the performance of LL-HARQ and LT-TCP under a uniform loss model. We note the low link latency and the low amounts of PFEC and RFEC wasted. Also note that transport performance is quite good with very few timeouts.

7.3.3 Performance of LT-TCP and LL-HARQ (4 hops)

We first look at the performance of TCP-SACK+LL-ARQ for the 4-hop topology shown in Figure 7.2. The high ARQ limit makes the link-residual loss rate negligible (0.65 %) even when the average PER is as high as 40%. However, the impact on the link-level latency is substantial, with the average *single-hop latency* going up from 10 milliseconds to nearly 100 milliseconds at 50% PER (see Table 7.3). This increases the *service time* per packet on the link, causing the link-level goodput to decrease dramatically as the average PER increases. Consequently, TCP-SACK goodput also reduces, as shown in Figure 7.8. In the worst case scenario, (average PER of 50%), the link protocol achieves neither low latency nor low residual loss rate. Clearly, a purely ARQ based link scheme in conjunction with TCP-SACK is not sufficient to deal with highly bursty errors over multiple hops. The transport goodput is 2.91 Mb/s at 30% PER but falls to almost zero beyond that (Figure 7.8).

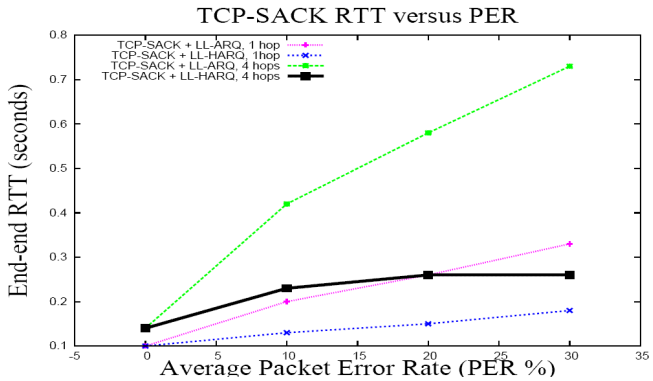


Figure 7.7: The end-to-end RTT experienced on a 1 and 4 hop topology with TCP-SACK as the transport layer and two different LL protocols is shown. It can seen that the end-to-end RTT is much lower using the LL-HARQ protocol at the link layer. This improvement is remarkable considering that the queuing delays are significant (since we have 10 flows).

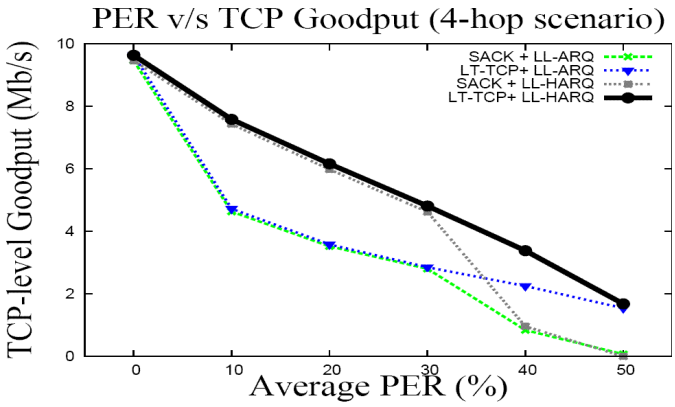


Figure 7.8: The transport-layer goodput for the 4-hop scenario is shown in this graph. With TCP-SACK as the transport protocol, the performance collapses beyond 30%. With LT-TCP however, the degradation in performance is linear, especially with LL-HARQ as the link protocol. LL-HARQ also leads to lower link latencies compared to LL-ARQ.

10 Flows, 4 hops	AVERAGE PACKET ERROR RATE					
PARAMETER	0 %	10 %	20 %	30 %	40 %	50 %
Receiver Goodput(Mb/s)	**9.63**	**7.58**	**6.16**	**4.81**	**3.38**	**1.68**
CI Bounds (Goodput)	[9.62,9.64]	[7.56,7.61]	[6.12,6.19]	[4.78,4.85]	[3.38,3.38]	[1.62,1.74]
Link level Residual Loss Rate	**0.00**	**0.00**	**0.01**	**0.08**	**0.61**	**3.99**
Link level Goodput	**9.93**	**7.81**	**6.35**	**5.02**	**3.77**	**2.41**
Average Link Latency (ms)	**11.08**	**14.69**	**15.65**	**17.73**	**19.36**	**20.08**

Table 7.2: 10 flows, 4 hop topology, ON-OFF Error Model: LT-TCP+LL-HARQ performance for different loss rates. The combination improves receiver goodput significantly with tight control on latency. LT-TCP overcomes the 4% residual link error rate.

We now examine improving LL-ARQ with LL-HARQ (using PFEC and RFEC). Table 7.3 shows the effect of using LL-ARQ and LL-HARQ with TCP-SACK on the link latency. With LL-HARQ, we achieve a much smaller average link latency of less than 20 ms even in the worst case (ON-OFF model,average PER = 50%). Moreover, this translates to a much smaller end-to-end delay experienced by the transport layer. Figure 7.7 shows the end-to-end TCP-SACK RTT for the two cases. In spite of significant queuing delays due to 10 concurrent flows which contribute to the latency per hop, LL-HARQ is able to significantly reduce the end-to-end delay seen by TCP. However even with LL-HARQ, the end-to-end TCP-SACK goodput drops rapidly as the link error rate increases. For example, at a PER of 30 %, the TCP-SACK goodput was 4.62 Mb/s (Figure 7.8) with a per-hop link residual loss rate of 0.09%. This small per-hop residual loss rate accumulates as we go over multiple hops, especially at these higher loss rates. In fact, in the worst case scenario (PER of 50%), the TCP-SACK connection collapses and the goodput goes down to zero. Consequently, we achieve very poor end-to-end TCP-SACK performance at high loss rates, suggesting that these residual loss rates must be tackled at the TCP layer.

We now switch from TCP-SACK at the transport layer to LT-TCP. However, an immediate question that arises is: Can the performance issues be solved at the transport layer alone (still using LL-ARQ as the link protocol)? We test the performance of LT-TCP+LL-ARQ to see if the changes at the transport layer alone will be sufficient. The performance in this case is better over 4 hops compared to

10 Flows, 4 hops	AVERAGE PACKET ERROR RATE					
PARAMETER	0 %	10 %	20 %	30 %	40 %	50 %
Average Link Latency (ms) (LL-ARQ)	10.83	32.18	45.63	59.64	80.19	98.80
Average Link Latency (ms) (LL-HARQ)	10.83	14.21	15.15	17.04	18.47	19.01

Table 7.3: 10 flows, 4 hop topology, ON-OFF Error Model: This table shows the link latencies for the LL-ARQ and LL-HARQ protocols with TCP-SACK. It can be seen that the average link-latency for LL-HARQ is much smaller as compared to LL-ARQ.

using TCP-SACK when using either of the two link layer protocol versions (LL-ARQ or LL-HARQ). With LL-ARQ, the LT-TCP goodput improves to 1.54 Mb/s at 50% average PER (from near zero with TCP-SACK). But as expected, the change in the transport to LT-TCP does not yield an improvement to the average link latency (and is comparable to TCP-SACK+LL-ARQ).

To reduce the latency on the link and improve the robustness to high loss rates, we replace the link protocol from LL-ARQ to LL-HARQ. Table 7.2 shows the performance of this combination (LT-TCP + LL-HARQ) in detail. At the link-level, it is worthwhile noting that at an average PER of 50%, the link residual loss rate is about 4%. Even with this, the four-hop end-to-end goodput with LT-TCP is quite significant (1.68 Mb/s), compared to TCP-SACK goodput of near 0. With the average link latency reducing to 20 ms with LL-HARQ compared to LL-ARQ's latency of 100 ms at a PER of 50% (Table 7.3), we find that the improvements by the transport (LT-TCP) protocol to goodput and link (LL-HARQ) protocol to latency complement each other. Thus, these modifications help achieve our key objectives of low residual loss rate on the link, low average latency (average number of transmission attempts is less than 2), high link and transport-level goodput.

Figure 7.9 shows the link-level goodputs obtained for the single-hop topology shown in Figure 7.1. In the case of link-level goodput, we determine the impact of the LL-HARQ scheme by comparing the performance of TCP-SACK with LL-ARQ and LL-HARQ (see Figure 7.9). The replacement of LL-ARQ by LL-HARQ clearly increases the achieved goodput consistently.

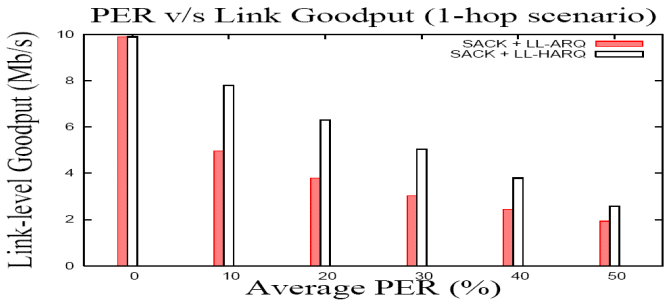
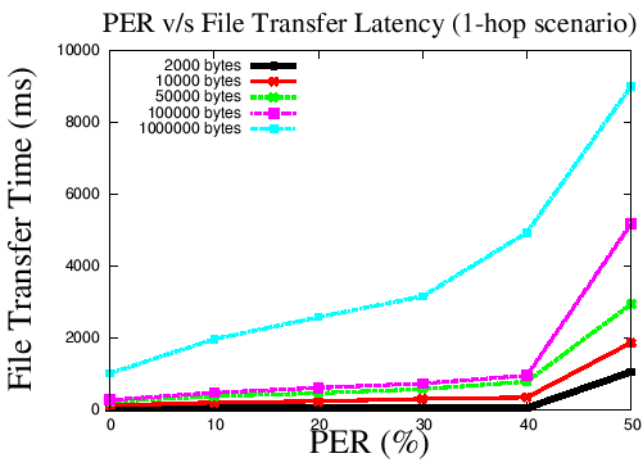


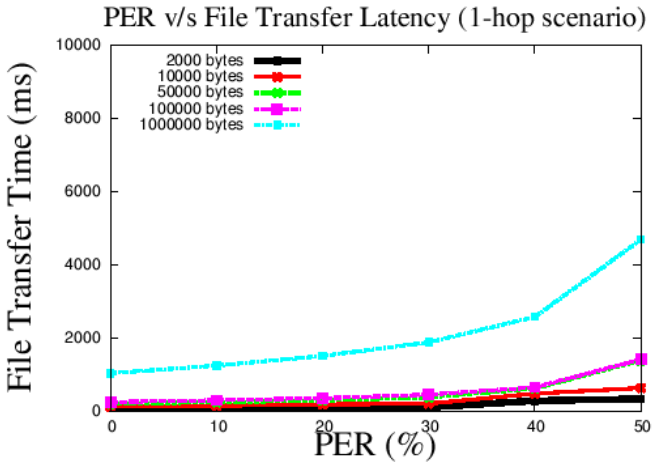
Figure 7.9: Link-level goodput for the single hop topology.

7.3.4 Impact on File Transfer Latency

We now look at the impact of two key schemes (TCP-SACK+LL-ARQ versus LT-TCP + LL-HARQ) schemes on the file transfer latency. We define file transfer latency to be the interval between the sending of the first packet of the file and the receipt of the ack for the last packet at the transport layer. We consider the single-hop topology shown in Figure 7.1 suffering from a bursty Gilbert error model. We focus on the transfer of relatively small files, as these transfers are likely to be the ones that are latency sensitive.



(a) TCP-SACK + LL-ARQ



(b) LT-TCP + LL-HARQ

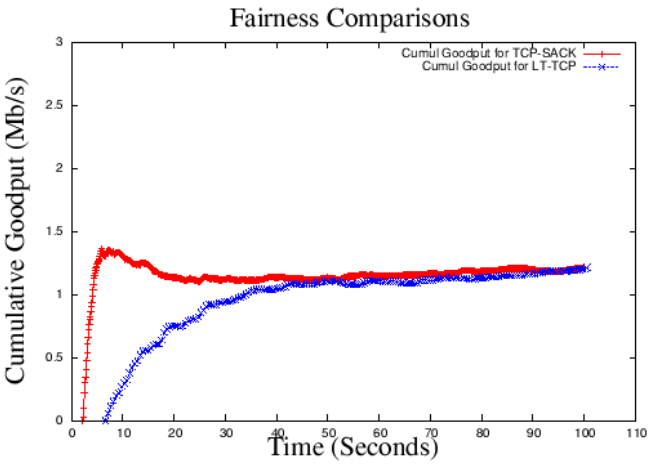
Figure 7.10: The link-level goodput obtained and the latency impact are shown. We consider files of 2K, 10K, 50K, 100K and 1M bytes.

Figures 7.10(a) and 7.10(b) show the file transfer latencies for the TCP-SACK with LL-ARQ and LT-TCP with LL-HARQ respectively. While the combination of LT-TCP+LL-HARQ is better under almost all cases, it is interesting to see that with small sizes, performance of TCP-SACK+ LL-ARQ is slightly better. However, this is true only when the residual loss rate is very low (almost 0%) which is the case when the average individual link PER is 40%. The ARQ (or HARQ) mechanism improved performance with TCP-SACK+ LL-ARQ are a) For very small file sizes, with TCP-SACK the number of TCP packets sent is small, namely 2, versus 10 is capable of exporting an error free channel under these cases. The reasons for the packets sent (due to granulation) with LT-TCP+LL-HARQ. b) Residual loss rate is negligible requiring no retransmissions at TCP layer. As the file size increases, however, LT-TCP+ LL-HARQ significantly outperforms TCP-SACK+LL-ARQ. This is particularly true under extreme conditions when the average PER is 50%. For example, in the 100K file size case, LT-TCP+LL-HARQ has one-third the latency as TCP-SACK+LL-ARQ. In the 1M file size case, the improvement in file transfer latency with LT-TCP + LL-HARQ over TCP-SACK+LL-ARQ is a factor of two.

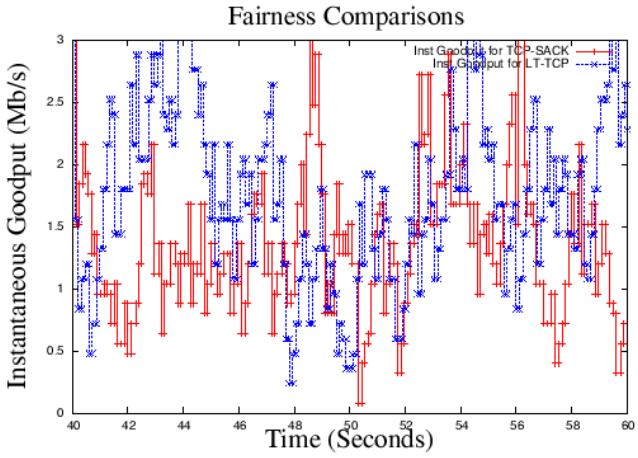
7.3.5 Fairness Among LT-TCP and TCP-SACK Flows

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 (see Figures 7.11(a) and 7.11(b)). 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

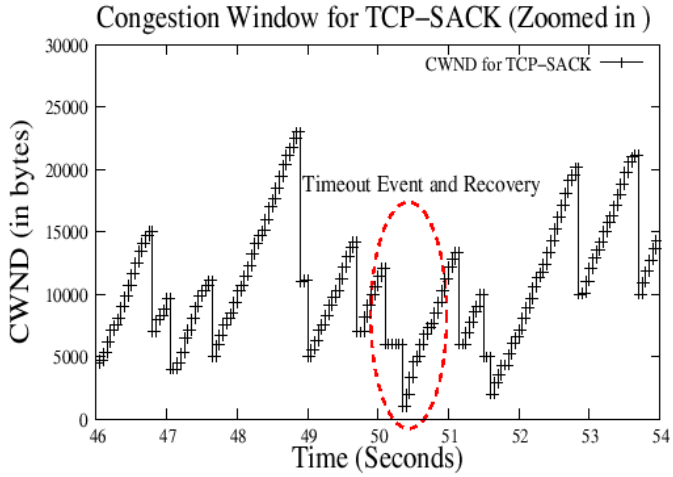


(a) Cumulative Goodput

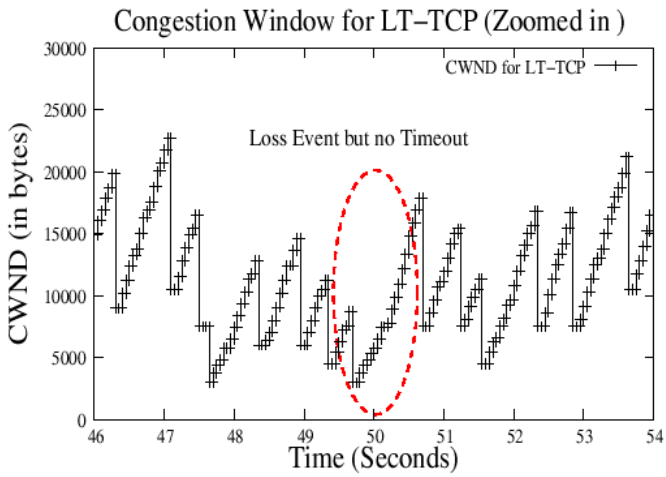


(b) Instantaneous Goodput

Figure 7.11: **In this scenario we have 5 connections of each type operating in a lossless environment. 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.**



(a) Congestion Window (TCP-SACK)



(b) Congestion Window (LT-TCP)

Figure 7.12: **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.**

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 just after this loss interval. The cumulative goodputs (measured as the goodput from the start of the simulations) in Figure 7.11(a) converge to give both flows an almost equal share of the bandwidth. The instantaneous goodput (measured for every 100ms window) in Figure 7.11(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 Figure 7.12(a) 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 Figure 7.12(b)).

In summary, 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.

This chapter evaluated the performance of the LL-HARQ and LT-TCP protocols over a variety of lossy scenarios. We looked at the performance over single and multi-hop links for Uniform, Gilbert and Disruption loss processes. We studied the trade-off between ARQ persistence and link goodput at the link layer. We also saw the impact over multiple hops. Support from the transport layer through LT-TCP enhancements is needed to provide good performance over multiple hops. The impact on file-transfer latency was also studied and it was shown that LT-TCP support can reduce this latency. We also saw that LT-TCP does not create unfairness towards non-LT-TCP flows such as TCP-SACK that may be operating concurrently.

CHAPTER 8
Performance Study of Link and Transport Protocols in Airborne Networks

8.1 Introduction

Recently, there has been a great deal of interest in the development of viable airborne network (AN) architectures for both defense and commercial applications. The design of an airborne network differs from that of traditional (e.g. Internet) networks because the dynamics of airborne networks cause a number of difficult conditions. In this work, we focus on link disruption caused by movement and changing orientation of network nodes/platforms. This includes wing and body blockage, terrain blockage, range limitations, atmospheric turbulence and weather effects. The focus of this chapter is mitigation of disruption of links in the AN. In this work, we leverage a combination of two technologies, a transport protocol LT-TCP (see chapter 4) and a hybrid link-layer FEC/ARQ protocol, LL-HARQ (see chapter 5). As discussed earlier, LT-TCP is an enhancement to the transport layer protocol TCP, that uses proactive and reactive FEC to provide higher goodput on unreliable links. LL-HARQ is a link- layer protocol that provides improved reliability on a link-by link basis via FEC and retransmission of packets. We investigate the performance of these approaches, on realistically modeled wireless links using actual link data gathered from AN experiments. We compare this performance to that of traditional link and transport approaches. Our results show that our proposed solutions greatly enhance network goodput for airborne networks where disruptions occur frequently when compared to traditional networks. In this work, we show that a combination of transport layer and link layer approaches can successfully address link disruption in wireless network contexts.

There are numerous examples of the use of wireless links for information sharing in the DoD context. These include satellite relays, UAV/aircraft relays, *ad hoc* microwave relays and ground stations/vehicles. Of particular interest in our work is the context of Airborne Networks (AN) for the DoD. These networks are often

mission-based, which means they are constructed to support a particular mission and must handle inherent dynamics of the mission. Unlike more static commercial wireless networks, these networks 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 potentially available capacity on links may be unsatisfactorily low. Link disruptions occur on a number of time-scales: small (bit or packet erasures), medium (interference, jamming, capture effect) and long (persistent jamming, channel impairments, longer-term link failures). In addition, even functional links can have variable end-to-end performance in terms of capacity, delay and packet loss.

In this chapter, we show the performance of our proposed protocols LT-TCP and LL-HARQ in the context of an AN. We use data captured from flight test experiments conducted by engineers at MIT Lincoln Laboratory as a basis for simulations of combinations of link and transport protocols using the ns-2 simulator. The collected data is formed into realistic link error processes over which we run a number of simulations using LT-TCP and LL-HARQ. Results show that the technology we have proposed achieves good performance and is appropriate in a dynamic AN context.

8.2 Airborne Network Experimental Design

Figure 8.1 illustrates our concept of the AN and the components involved. The figure shows that the network is comprised of a heterogeneous set of links. Shown in the figure is the concept of the Small Combat Network (SCN) which is the network comprised of a single airborne battle unit. This network includes wireless omni-directional links, as well as directional links connecting the SCN to backbone infrastructure. The figure also shows that the backbone is comprised of a number of high-rate directional wireless links. Also shown are satellite communication links, which are generally fixed geographically.

In such environments, link performance variability is the norm: a transport protocol such as TCP sees variable capacity and unpredictable *residual* packet erasure rates (PER). Seamless communication under such conditions requires tolerance

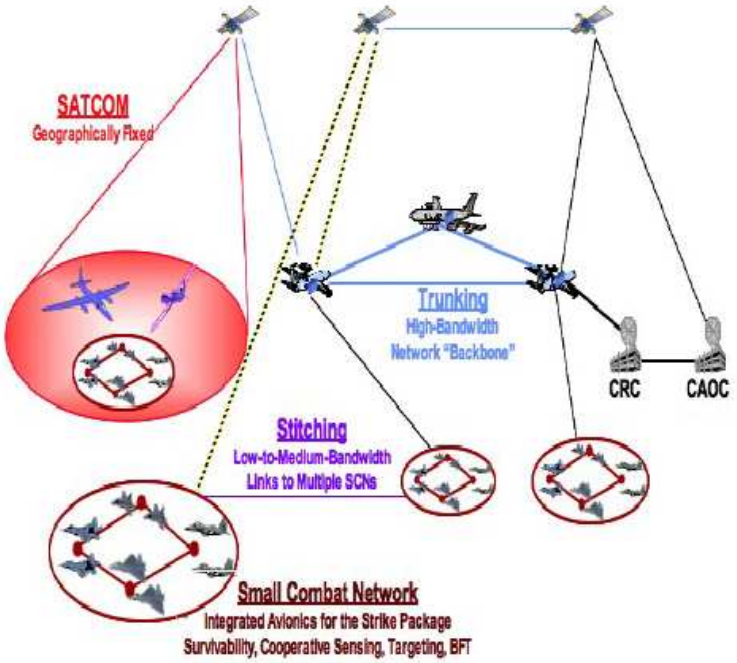


Figure 8.1: A small combat network (SCN) in operation.

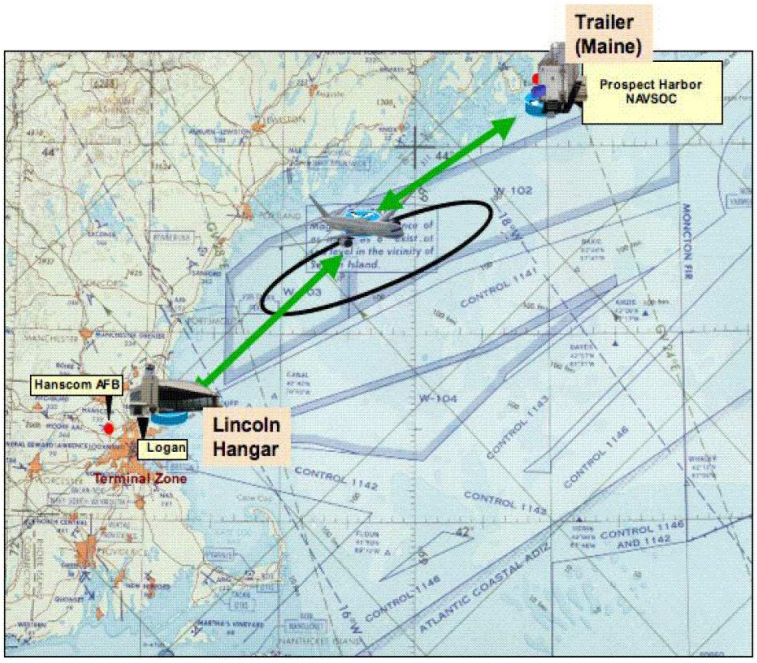
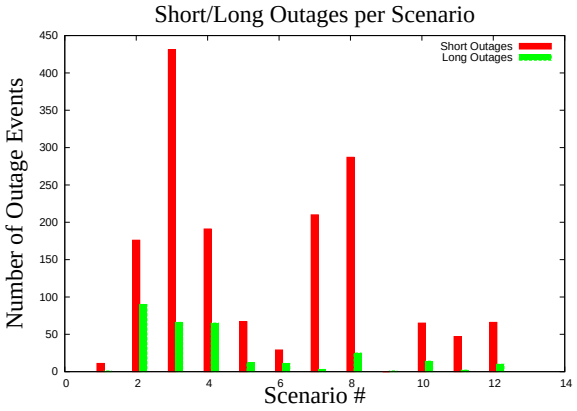
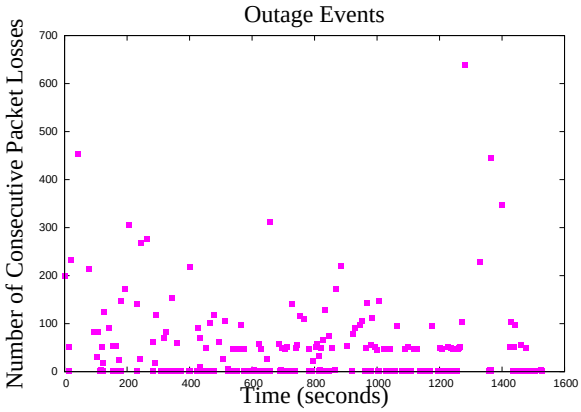


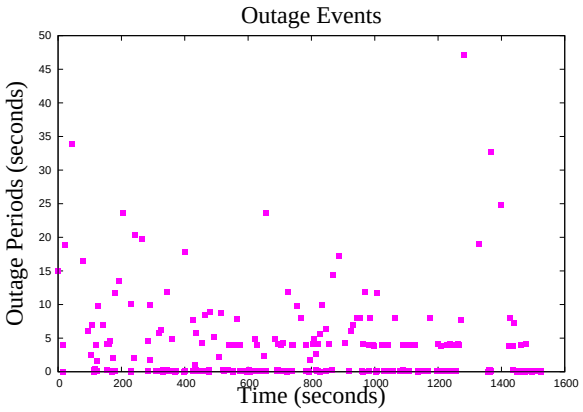
Figure 8.2: The figure shows the flight path of the planes involved in the gathering of the data traces. Also shown are the ground locations; one at Maine and one at MIT-Lincoln Labs in Boston



(a) Frequency of Occurance of Short and Long Outages



(b) Distribution of Outages (by Number of Packets Lost)



(c) Distribution of Outages (by Length of Outage)

Figure 8.3: The number of short and long outage events for the scenarios under consideration is seen in Figure 8.3(a). On the right, we see (for a specific scenario lasting 1600 seconds), the distribution of outage times. In particular, we note that while most of the outages are relatively short, the number of long-lived outages is non-trivial.

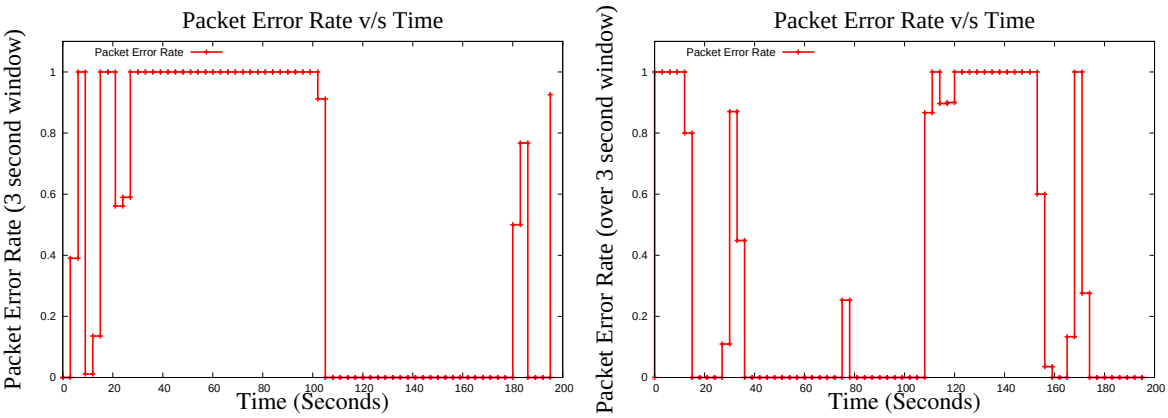
of such performance variability, especially packet erasures. Uniform residual error rates of 5% are severe enough to cause TCP variants such as TCP-SACK to collapse if they last for some time. At the link layer, packet losses are handled either by retransmitting them at the link-layer or by using techniques such as retransmissions or forward error correction (FEC) or by using a hybrid technique. Retransmissions lead to variable delay at the transport layer. Moreover, performance is drastically affected when the link suffers from periods of outages (when the loss rate is 100%). This translates to burst errors at the transport layer which leads to the collapse of the transport connection. The focus of this chapter is to test the performance of LT-TCP and LL-HARQ under such realistic and error-prone conditions. The high and variable packet error rate and periods of disruptions provide a stressful environment to test our solutions.

8.2.1 Flight Test Description

In August of 2006, engineers at Lincoln Laboratory conducted a series of five flight tests, involving two ground stations and a Boeing 707 aircraft interconnected using a number of communication radio types. One of these was a Common Data Link (CDL) radio which is a wireless directional link type, which we used for our experiments. The aircraft was flown in a realistic trajectory off the coast of Maine, and established connections with two ground stations several hundreds of miles away. Data was collected during the flights, and recorded for later analysis. This data included link status, packet send and receive logs, platform positional data, and application-based data. All the links, specifically the CDL link, were subject to disruptions and variable packet error rates, which were reflected in the collected data. The data is thus similar to what would be seen in a real-world AN scenario and its purpose is to serve as a basis for network design analysis.

For the experiments in this chapter, we have used this flight data collected to create a realistic "link error" process. In order to do this, we correlate the packet-level data on the send and receive side of a given link to determine the times at which link was up or down. From this analysis, we form a link status trace that we can use as input to our simulations involving LL-HARQ and LT-TCP. One of the

strengths of our work is the realism of the scenario produced by using actual flight data.



(a) Sample Scenario 1

(b) Sample Scenario 2

Figure 8.4: We process the trace data using a windowing algorithm to get wireless link-loss behavior as shown. These are then used as input to the ns-2 simulations to test the LT-TCP and LL-HARQ protocols.

8.2.2 Processing of Trace-data

This subsection provides a brief description of the generation of the link up-down process from the raw data. The experiment involved four CDL transceivers distributed among the platforms. The locations of the transceivers are Hanscom Base (in Boston), Trailer (in Maine), Airplane-top and Airplane-belly. Connections were made between both the Airplane-top and Airplane-belly nodes to the two ground bases. The connections are bidirectional, and IP packet data were recorded at each transceiver using the *tcpdump* program. From the packet-based *tcpdump* data traces, we construct the link error process as shown in Algorithm 5. We consider outages that are less than 5 seconds in length to be short outages and those over 5 seconds in length to be long outages.

The output of the processing is a link up/down process which is subsequently used as input to the network-layer simulations of LT-TCP and LL-HARQ. These processes are generated for each pair of transceivers (i.e. each link) and can be as

Algorithm 5 Trace Data Processing Algorithm

1: For all the packets in the *receive* file, hash the fields in the IP and TCP headers and store them.
2: Similarly, hash the packets in the *send* file. Packets that have a hash match have been successfully received. Packets that do not have a hash match are assumed to have been lost on the link.
3: This approach gives us loss/success statistics on a per-packet basis.
4: The result is a series of packet successes (designated '1') and packet losses (designated '0'). Periods of '0's marks outages. (In our processing, we see outages that last from a few milliseconds to tens (even hundreds) of seconds).
5: This sequence of '1's and '0's is converted into a link error process by processing it further.
6: This data is acted upon by the windowing technique described below to create wireless loss models that are used as input to the simulator.

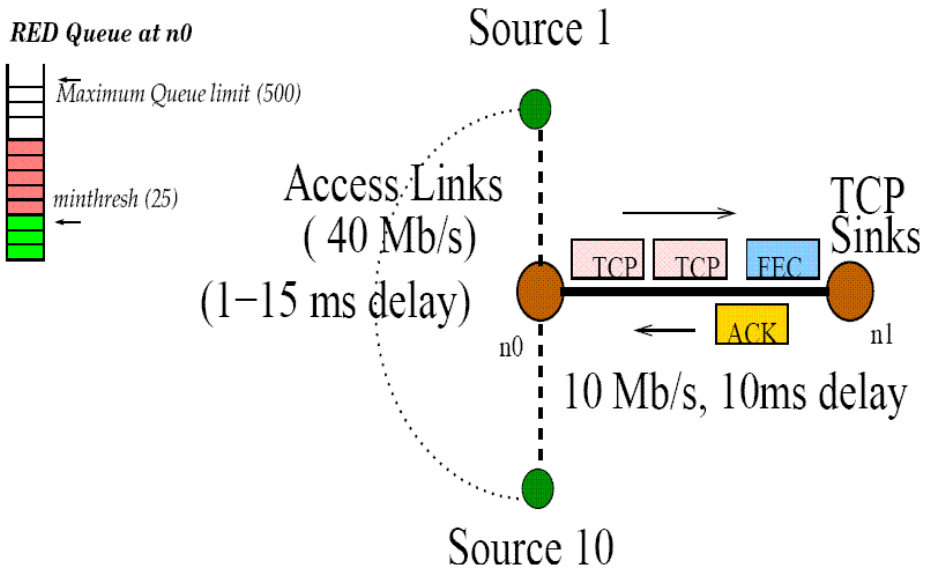


Figure 8.5: Test Configuration: The figure shows the topology used to test the developed protocols and compare them against baseline protocols. We use a 1 hop topology with an abstract lossy link.

long as the flight test itself. We segment the large traces into intervals of 100 seconds each and categorize them into short and long outage scenario types. Figure 8.3(a) shows that there is a mix of long and short outages in the traces over the 12 link traces. Figure 8.3(b) shows that the distribution of outage lengths ranges from 0 to 50 seconds over the course of the flight for a particular link. Note that short outages are more prevalent than longer, as expected. Figure 8.3(c) shows that the distribution of the number of packets lost in an outage.

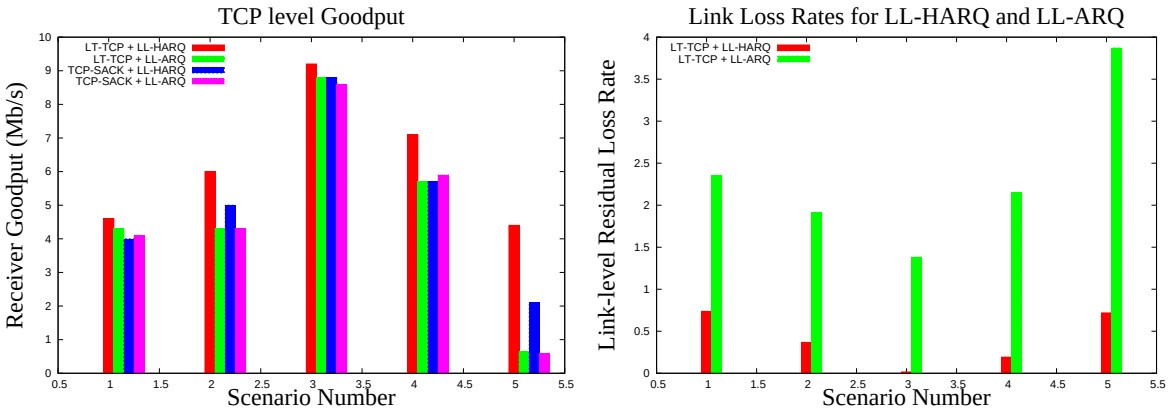
Generation of Error Models Using Windowing Apart from the outage instances, we can also look at the data and model it to represent a lossy link. The approach we choose here is a *windowing* approach where we average the loss rates encountered on the channel using a non-overlapping window of 3 seconds. Figures 8.4(a) and 8.4(b) show two samples of the wireless link error processes generated.

8.3 Results

This section tests the transport and link-layer protocols developed earlier in chapters 3, 4 and 5 with a set of targeted ns-2 based simulations. The topology used is as shown in Figure 8.5. At the transport layer, we have the option of using either TCP-SACK or LT-TCP while at the link layer, we can use either LL-ARQ or LL-HARQ. We test all four combinations below. LL-ARQ is the link-layer protocol that has no FEC support but has a high number of ARQ attempts (see chapter 7).

In Figure 8.6(a), we see that for all the scenarios considered, the combination of LT-TCP + LL-HARQ achieves the best transport goodput. It is also to be noted that the combination of TCP-SACK + LL-HARQ also performs well which points to the performance of LL-HARQ being crucial in these extreme scenarios where we not only have high and varying loss rates but also periods of outages. During the outage events, LL-HARQ's outage-detection and mode-switching kick in. This ensures that the residual loss rate exported to the upper layer is small. This can be seen in Figure 8.6(b).

In summary, we see that LL-HARQ with its ability to detect and counter disruptions and outages can significantly lower the residual loss rate that is seen



(a) (b)

Figure 8.6: Transport Goodput and Link residual loss rates for combinations of transport and link protocols are shown here for trace-driven wireless loss scenarios. Note the effectiveness of LL-HARQ in exporting a low residual loss rate even under high and variable losses including outages. LT-TCP + LL-HARQ achieves the best transport goodput among the various options.

by the transport layer. Moreover, since LL-HARQ uses far fewer retransmissions, the link latency is also small. In the cases where the exported residual loss rate is significantly large enough that TCP-SACK cannot deliver good performance, we see that replacing TCP-SACK with LT-TCP can help recover the performance. The combination of LT-TCP + LL-HARQ this yields synergistic benefits that can deliver good performance even in the most volatile of conditions.

8.4 Conclusion

In this chapter, we looked at real world traces of airborne links collected by MIT-Lincoln Labs from test flights between Boston, MA and Maine. We initially looked at the loss behavior of the wireless links to see the nature of outages and loss rates. We then modeled these links in our simulator using appropriate windowing techniques. We then tested LT-TCP and LL-HARQ over these models and saw how the LT-TCP and LL-HARQ improve the performance over TCP-SACK and LL-ARQ. In the next chapter, we will conduct additional measurements in wireless

LAN environments to study the nature of current link and transport protocols under noise-induced packet losses. We also study the interactions that happen between the two layers.

CHAPTER 9
Performance Study of Link and Transport Protocols in Noisy Wireless LAN Environments

9.1 Introduction

As discussed earlier (in chapters 1 and 2), the rapid deployment of broadband wireless systems such as 802.11 Wireless LANs (WLANs), 802.16 wireless broadband and neighborhood area wireless networks raises expectations of high end-to-end performance. As the demand for broadband connectivity increases, both cellular and meshed networks will play a role in last-mile wireless distribution networks. Current metro-WiFi planned deployments (e.g. San Fransisco, Google Wifi, AT&T Metro Wifi etc..) and organic community wireless deployments fit this model. It is well-known that wireless links have high, bursty and variable *raw* error rates due to atmospheric conditions, terrestrial obstructions, fast and multi-path fading, active interference and mobility[16]. However, for TCP, what matters is *residual packet erasures* and delay behavior after PHY and LINK layer mitigation has been completed [50]. TCP is exposed to residual error rates which is defined as the error rate *subsequent* to the link layer's error protection mechanisms.

We dig deeper into the sources of residual erasures in networks with 802.11a based access (last hop) links. Our focus is on the interaction between mechanisms at the 802.11a MAC layer and the transport layer in response to noise-induced packet corruption. In particular, we ask: "*Can* MAC *and transport protocols effectively deliver a significant proportion of the raw bit-rate available at the physical layer to the application in a multi-user environment prone to high raw loss rates?*" We investigate the performance of 802.11a MAC and TCP performance in this context. The link-level transmission (assuming rate-adaptation) and propagation times are small enough in LANs to allow multiple retransmission attempts. However, the utility of persistent ARQ is affected negatively due to delays between successive ARQ retries.

In this chapter, we investigate the nature of real wireless links by conducting

Figure 9.1: Open-Access Research Testbed for Next-Generation Wireless Networks: (ORBIT)

experiments on Open-Access Research Testbed for Next-Generation Wireless Networks (ORBIT)[8]. ORBIT consists of a network of 400 wireless nodes that can be remotely controlled to run experiments. One of the attractive features of ORBIT is the ability to load an operating system of choice (we use GNU/Linux) on the nodes and to use a open-source wireless driver released under the GNU license. This enables us to modify the driver and the operating system kernel to suit our interests.

Our objectives in this measurement study are as follows:

1. To study the nature of wireless losses in the presence of varying levels of noise. We wish to see the impact of noise on raw link loss rates, residual loss rate experienced by the transport layer and the throughput/goodput obtained at each layer and the overheads and penalties incurred.

2. We also wish to study the impact of high ARQ (retransmissions) at the link layer on the raw and residual loss rates. We can then in turn see the impact of the loss rates on the link and transport goodput.

3. While high ARQ can reduce the residual loss rates, the link latency can be increased. The latencies experienced on the link can be high and variable. This can not only have an impact on applications such as streaming video/ VoIP calls etc. but also interact negatively with transport layer mechanisms.

Delay and latency spikes can lead to spurious timeouts at TCP [98] which can have the effect of keeping the link idle leading to lower than expected throughput/goodput.

The rest of the chapter is organized as follows. Section 9.2 details the experimental setup on the ORBIT testbed and our results and insights from the measurements. Section 9.3 tests the proposed protocols using the ns-2 simulator. Section 9.4 concludes the chapter.

9.2 Experimental Setup

In this section, we look at the experiment setup and discuss the results and insights. We chose the 5.18 GHz (802.11a) frequency range so that we would not encounter interference from 802.11b wireless APs that are present in the rest of the building in which ORBIT is housed. First, we present some background on the experimental setup and discuss the changes made to the wireless driver and operating system to gather statistics on the metrics outlined below.

9.2.1 Background

In this section, we will provide an overview of the wireless driver used as well as the experimental parameters.

Changes to the Wireless Driver: Of the 400 nodes on the ORBIT testbed, we use the ones that have the Atheros AR5212 wireless cards. We use an open-source driver titled *madwifi* to control the Atheros chip-sets[1]. The *madwifi* wireless driver is based on the *net80211* wireless stack in the Linux kernel. The *net80211* stack provides a generic interface to access the kernel functionality and the *madwifi* driver provides access to the Atheros chip-set hardware to control it. The *madwifi* driver uses a binary-only Hardware Abstraction Layer (HAL) to control the low level functions of the chip-set. The HAL is released in binary-only from by Atheros since certain functions such as transmit power settings under certain frequency ranges are prohibited by FCC regulations.

Our main focus has been in making changes to the *madwifi* driver by the addition of various statistics and metrics gathering. These include, among other

things:

1. Making the ARQ (number of retransmission attempts per packet) user-settable. By default, this is 11. This makes it possible to study the impact of varying the ARQ on the link and transport performance. For the work presented in this chapter, the ARQ persistence is set to 14 (total of 15 attempts per packet).

2. Measuring the number of data/control packets sent on the channel and number of such transmissions lost in the face of noise. This gives us the raw loss rate on the channel.

3. Measuring the number of data packets sent by the transport layer to the link layer and the number of such packets that were not delivered even after the ARQ limit was reached. This gives us the residual loss rate on the channel.

4. Other statistics of interest such as transport and link throughput and goodput, distribution of losses for different ARQ tries etc.

The *madwifi* driver is not part of the core kernel release and is installed as a loadable module into the running Linux kernel in our experiments. One advantage of using a loadable module is that changes made to the driver does not need a kernel recompilation and a reboot. Only the module needs to be unloaded, recompiled and loaded again.

Changes to the Operating System Kernel: One of the advantages of the ORBIT testbed is that it is possible to "reimage" the nodes with an operating system of one's choice. For the experiments we performed, Linux kernel 2.6.12 was used. The *madwifi* driver is not part of the official Linux release and must be compiled separately and inserted into the running kernel.

Apart from the link layer where changes to the *madwifi* driver were made, we were also interested in the transport layer protocol (TCP) to see how the link layer loss characteristics impact TCP mechanisms. To this end, various changes were made to the TCP layer. The TCP layer is part of the core kernel (it is not a module like the *madwifi* driver). Therefore, changes to the TCP layer involve a recompilation and installation of the kernel followed by a reboot.

At the TCP layer, the following metrics were gathered:

1. Number of packets and bytes of unique (new) data that were sent, number of packets and bytes that were retransmitted and the header overhead in each.

2. The times when TCP timed out and the sequence number of the packet timing out.

3. The times when TCP entered Fast Retransmit and packets retransmitted as a consequence.

4. RTT samples and the behavior of the smoothed RTT (SRTT) and the Retransmission timeout (RTO).

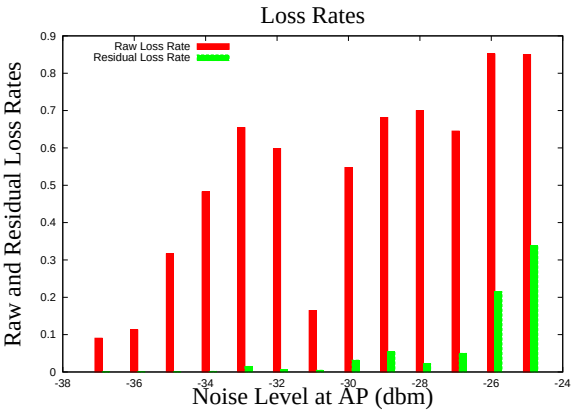
5. Behavior of the congestion window as it changes with time. We also see the number of packets that are outstanding (sent but not yet acked).

The Linux kernel has a granularity of 1 ms. RTT samples are gathered with a granularity of 1 ms. The minimum RTO is 200ms, the maximum RTO is 12 seconds and the default RTO value is 3 seconds.

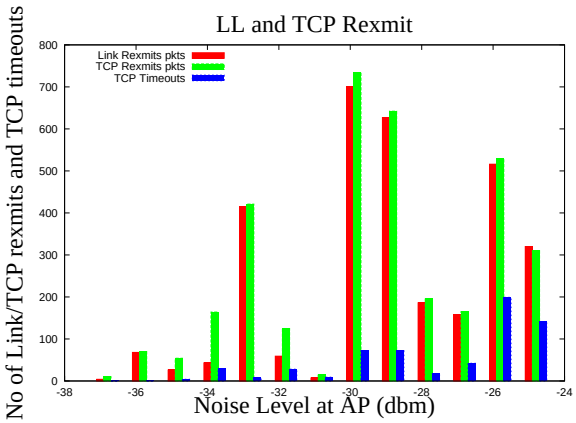
9.2.2 Experimental Results

Our main setup uses three nodes: one acting as a wireless AP, one as a client and a node near the client acting as a sniffer (see Figure 9.3). We run *iperf* in server mode at the AP and in client mode at the client for 60 seconds and gather our data. Apart from running *tcpdump* on the sniffer, we use a module (*tcpprobe*) on the client to collect TCP layer information. The wireless card on the sniffer is operated in a promiscuous mode and *tcpdump*[11] is used to capture packets. The *wireshark* tool is used to decode and understand these packets [14]. By correlating the information gathered at the link layer (*madwifi* driver), TCP layer and the *tcpdump* data traces, one can get an idea of what interactions happen between the link and transport layers. In particular, we are interested in seeing what happens to packets that have timed out at TCP.

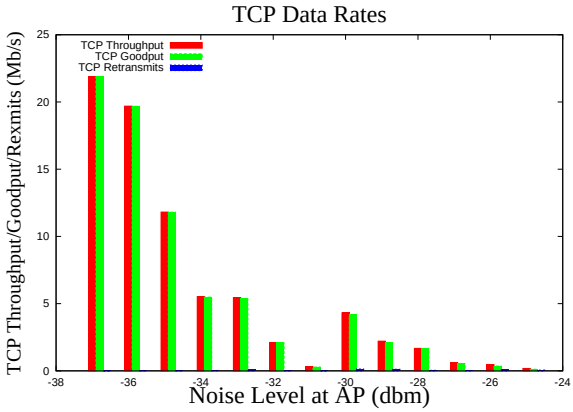
To this setup, we add varying amounts of noise through the Noise Injection Subsystem available on the testbed. The amount of noise injected was between



(a) Loss Rates



(b) Retransmissions and Timeouts



(c) Statistics of TCP performance

Figure 9.2: For noise level -30dBm, sub-figure 9.2(a) shows the raw and residual loss rates seen on the channel. We see that the raw loss rate can be quite high which leads to a significant residual loss rate seen by TCP even with a high ARQ limit. Sub-figure 9.2(b) shows the number of retransmitted packets at the two layers and the number of TCP timeouts. Sub-figure 9.2(c) shows the transport-layer throughput, goodput and retransmitted bytes. It should be noted that as the noise level increases, the performance falls gracefully.

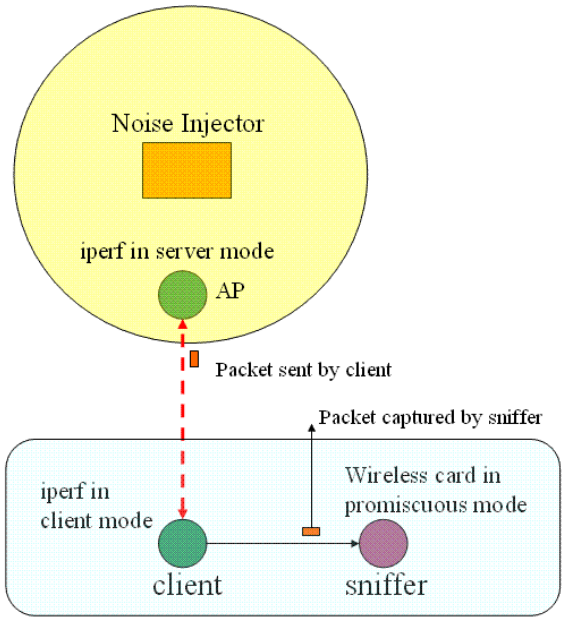
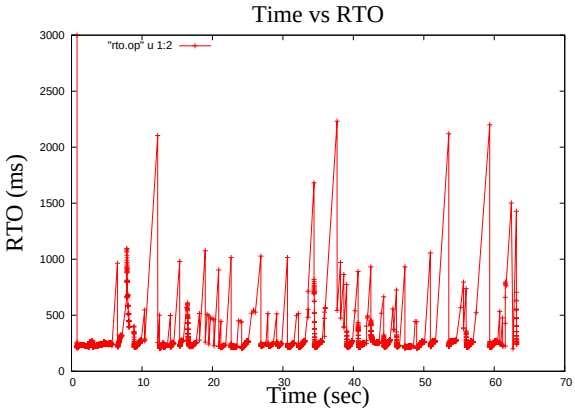


Figure 9.3: Experimental Topology: The topology used to study the performance of the link and transport protocols and the interactions between them is as shown.

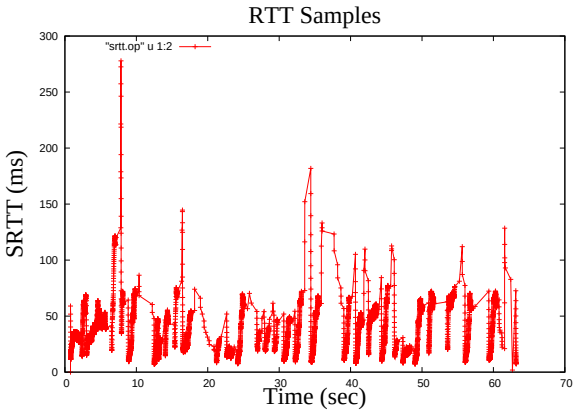
-25dBm which provides a highly noisy environment to -40dBm which is almost a noise-free environment. By using varying levels of noise, we can study the behavior of the link and transport protocols in the face of packet loss and the interactions between the two layers.

9.2.3 Analysis of Loss Rates

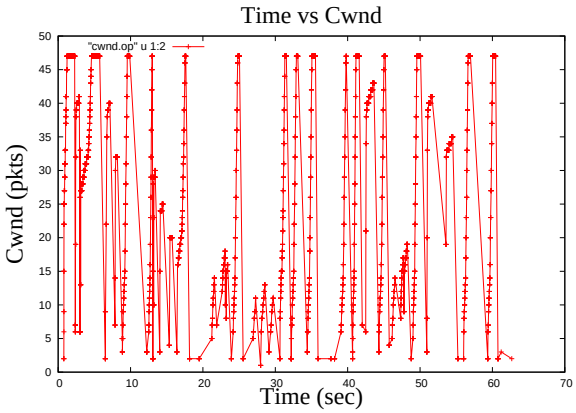
Figure 9.2(a) shows the raw loss rates on the channel at different noise levels. The figure also shows the residual loss rate that is exported to the transport layer. This residual loss rate is the loss rate that is seen by TCP subsequent to the 15 transmission attempts. It can be seen that the raw loss rate can rapidly increase but the performance (throughput and goodput) obtained degrades relatively gracefully. Figures 9.2(c) show that the performance drop, contrary to conventional wisdom is not a "cliff" but is more graceful. However, we can see that as the residual loss rate goes up beyond 5% (noise level -29dBm), the TCP performance drops rapidly. TCP is thus susceptible to even relatively loss residual loss rates even under conditions of very small round trip times. This susceptibility will only increase as the round



(a) Retransmission Timeout



(b) Smoothed RTT



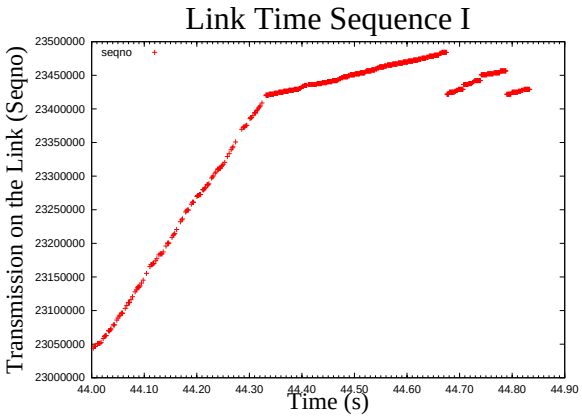
(c) Congestion Window

Figure 9.4: Window and RTT plots (noise level -30dBm): These three figures show the behavior of the RTO, smoothed RTT (SRTT) and the congestion window in packets for noise level -30dBm.

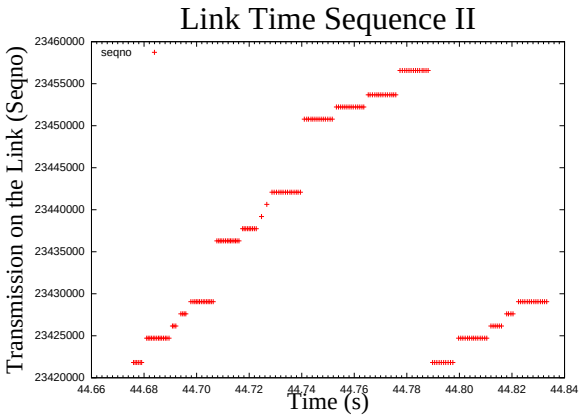
trip time increases.

9.2.4 Analysis of Timeouts

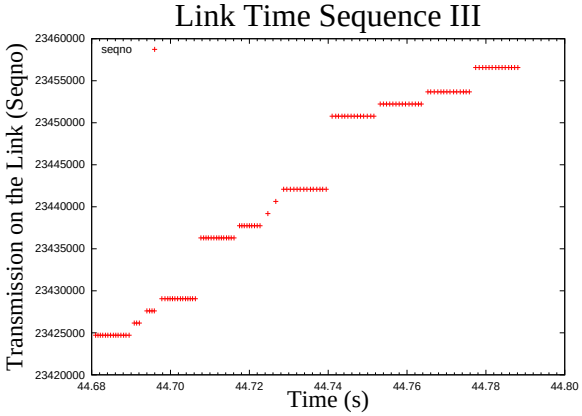
We now look at the instances of timeouts in our experiments. As the noise level on the channel increases, clearly the raw loss on the channel goes up and consequently, the residual loss rate that is exported to the upper layer also goes up. In spite of having a high number of ARQ attempts (1 original attempt + 14 retransmission attempts), there is evidence that some of the timeouts that occur are not simply due to the packet exceeding its link retransmission attempts but due



(a) TCP-Link Interaction - Time Interval: 44.0-44.9s



(b) TCP-Link Interaction - Time Interval: 44.66-44.84s



(c) TCP-Link Interaction - Time Interval: 44.68-44.80s

Figure 9.5: For noise level -30 dBm, we see the occurrence of a spurious retransmission and timeout. The first figure shows the transmission of packets around the period of interest and we then zoom into this period further. We see that the packet with seq no 23421817 is sent thrice. While attempt 2 is successful, the high latency incurred causes a timeout resulting in a third transmission of the packet.

to interactions with TCP. We now look at the instances of timeouts and categorize them based on their causes.

9.2.4.1 Timeouts due to packet loss

Most of the timeouts that we have observed are due to the fact that a packet that was given to the link layer by TCP could not be sent across the link because the ARQ limit of 15 was reached. In this case, if duplicate acknowledgments did not reach TCP (entire window was wiped out), then TCP is left with no option but to timeout and retransmit the packet. The vast majority of the more than 600 timeouts that we encountered over all the noise levels were caused by this (see Figure 9.2(b)).

9.2.4.2 Timeouts due to link latency and TCP interactions

Packets can also suffer from increased packet latency on the link if packets that are being serviced before it incur a large number of ARQ attempts and thus high link delay. Figure 9.5(a) shows the number of attempts taken by intermediate packets between the 3 transmissions of TCP packet with relative seq no 23421817. The initial packet transmission fails and after 15 attempts, the link layer gives up. However, duplicate acks at the TCP layer detect the need to send this packet and the packet is retransmitted. However, this retransmitted packet is stuck behind a number of packets waiting to be delivered at the link layer several of which take almost 15 attempts (and some are delivered). The effect of this is that though the retransmitted packet is successfully delivered over the link in 7 attempts, TCP has timed out and the packet is then sent across the link a third time (this time taking 11 tries). Figure 9.5(b) also shows the times taken by the intermediate packets. Note that since we have only 1 client, no exponential back-off is present. In the presence of a large number of nodes, exponential back-off would have exacerbated this effect. The retransmission timer RTO at TCP is 200 ms and is triggered by the large link latency seen by the second transmission attempt. This other cause of timeouts is harder to detect since it depends on protocol interactions between the link level driver and the transport protocol. Moreover, this effect manifests itself infrequently. It should also be noted that such interactions will be manifested more frequently as

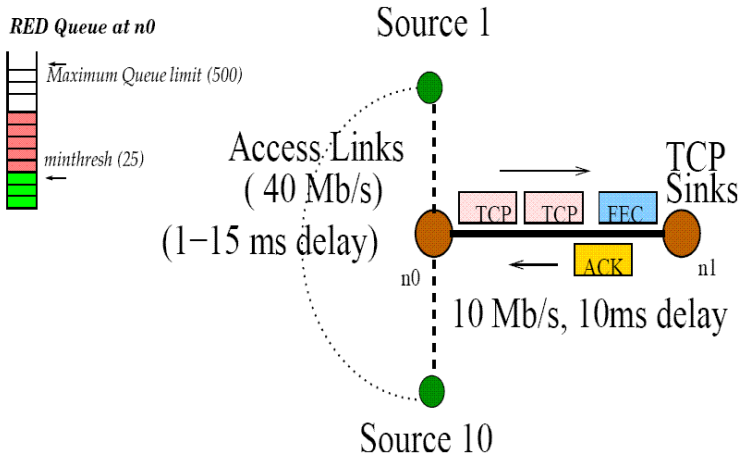


Figure 9.6: Test Configuration: The figure shows the topology used to test the developed protocols and compare them against baseline protocols. We use a 1 hop topology with an abstract lossy link.

the RTT increases (when we move from single hop to multi-hop scenarios) and as the bandwidth increases. Moreover, in the presence of multiple clients, exponential back-off between successive transmission attempts will exacerbate this effect.

In summary, we see that the performance of link and transport protocols degrades rapidly with increasing raw and residual loss rates. Moreover, this degradation will worsen with emerging scenarios such as multi-hop networks and metro-wide broadband which will present increases RTT and bandwidth. To obtain a good trade-off between link goodput, residual loss rate and link latency, it is important to design link protocols that can provide high goodput and low loss rate with low ARQ persistence. We now see how LL-HARQ can meet these performance objectives.

9.3 Performance Evaluation

This section tests the link-layer protocol developed earlier in chapters 3 and 5 with a set of targeted ns-2 based simulations. The topology used is as shown in Figure 9.6. We had outlined the design objectives for LL-HARQ in chapter 5. Below, we recap the LL-HARQ and LL-ARQ protocols to provide context.

10 Flows, Single-link	ERROR RATE					
PARAMETER	0 %	10 %	20 %	30 %	40 %	50 %
Goodput (Mb/s)	9.96	8.05	6.71	5.61	4.58	3.59
Throughput (Mb/s)	9.99	9.98	9.99	9.99	9.98	9.99
Residual Loss Rate (%)	0.00	0.00	0.00	0.00	0.00	0.00
Avg. Latency (ms)	10.97	13.83	13.26	13.80	14.67	15.05
PFEC Sent (Mb/s)	0.02	1.48	2.90	3.77	4.60	5.40
RFEC Sent (Mb/s)	0.00	0.39	0.36	0.59	0.80	0.98
PFEC Wasted (Mb/s)	0.02	0.61	1.04	1.03	1.02	1.00
RFEC Wasted (Mb/s)	0.00	0.26	0.23	0.33	0.38	0.39

Table 9.1: LL-HARQ Details: The table shows the various performance metrics for the LL-HARQ scheme. The residual loss rate exported to the transport layer in all cases is 0. Also, compared to the amount sent, the wastage in PFEC and RFEC is small.

9.3.1 Evaluation of Link Protocols: LL-HARQ and LL-ARQ

LL-ARQ is the baseline link-level protocol which has a pure ARQ mechanism (limit on the number of ARQ attempts set to 15) but without FEC support. To be conservative, LL-ARQ does not incorporate the typical back-off mechanisms between ARQ retransmissions. **LL-HARQ** is our proposed link protocol which has a limit at most one ARQ retransmission attempt and includes PFEC and RFEC mechanisms. In both variants, an incoming packet at the link layer is split into 20 fragments. With LL-ARQ, retransmissions are the same as the original transmission. With LL-HARQ, retransmissions consist of RFEC fragments. LT-TCP is the transport layer protocols used. We use a uniform loss scenario with the packet error rate ranging from 0 to 50%.

Figure 9.7(a) shows the transport layer goopdut obtained with LL-ARQ and LL-HARQ. We note that both LL-ARQ and LL-HARQ export a negligible loss rate to the transport layer. However, LL-HARQ manages to do this with just two transmission attempts (1 ARQ) which keeps the link latency low. LL-ARQ relies on a high number of ARQ attempts (15) to achieve a low residual loss rate which causes high link latency (see Figure 9.7(b)). Figure 9.7(c) compares the average number of timeouts incurred at the TCP layer. Table 9.1 shows the detailed performance

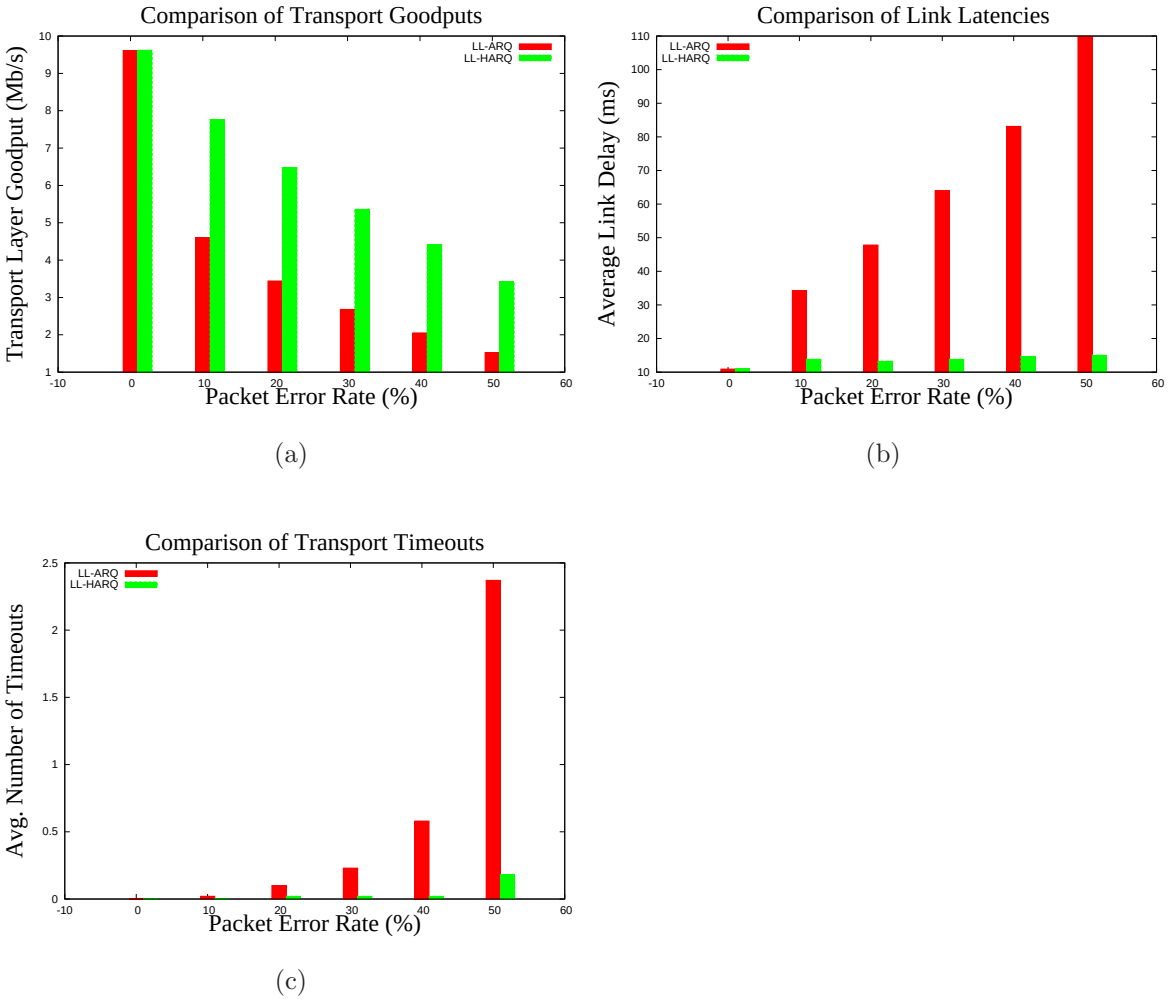


Figure 9.7: We compare the transport layer goodput and the per-packet average link latency for the LL-ARQ and the LL-HARQ link protocols with LT-TCP as the transport protocol. It can be seen that the transport goodput and the latency are much better with LL-HARQ. Both variants export zero residual loss rate to TCP but because LL-HARQ uses only 2 transmission attempts, the obtained trade-off is much better than with LL-ARQ.

metrics for LL-HARQ.

9.4 Conclusion

In this chapter, we studied the performance of current link-layer protocols and TCP on real systems. The experiments were performed on the ORBIT wireless testbed at Rutgers University. To facilitate our study, we made several changes to the TCP layer in the Linux kernel as well as the *madwifi* wireless driver.

We saw how the performance of the link protocol as well as the performance of the transport protocol are affected by the introduction of noise. High ARQ persistence can help bring the raw loss rate down at the link layer which is exported as small residual loss rate at the TCP layer. However, the high ARQ persistence extracts a cost in term of high link latency which leads to high and variable delays as measured at the TCP layer. Moreover, this can lead to subtle interactions between the link and transport layers such as spurious retransmissions and timeouts. At the link layer, the high raw loss rate leads to reduced throughput and goodput.

To achieve a favorable trade-off between goodput, residual loss rate and latency, we designed a link protocol that can deliver high goodput and low residual loss rate with just 1 ARQ retransmission. We compared the baseline LL-ARQ and proposed protocol LL-HARQ using simulations on the ns-2 simulator. We saw how the trade-off among goodput, residual loss rate and average link latency can be improved dramatically with LL-HARQ as compared to LL-ARQ.

Acknowledgments: The authors would like to thank Prof. Dipankar Raychoudhari and Prof. Ivan Seskar of WINLAB,Rutgers University for the use of the ORBIT testbed and for technical discussions.

CHAPTER 10
Conclusions and Future Directions

10.1 Summary and Conclusions

This thesis addressed the performance problems experienced by link and transport protocols over network paths that include lossy links. It is well known that TCP performance degrades sharply beyond an end-to-end loss rate of 5%. At the transport layer, we have proposed a highly loss-tolerant TCP protocol (LT-TCP) using an adaptive, end-to-end hybrid ARQ/FEC reliability strategy exploiting ECN for congestion detection. To complement this transport protocol, we proposed a link-level protocol (LL-HARQ) that meets the objectives of exporting a low residual loss rate while maintaining low latency and high link goodput even under scenarios when the link experiences disruptions.

We considered an abstract, lossy link and developed link-level mechanisms designed to meet the above goals. Our building blocks included an accurate loss rate estimation technique, adaptive segmentation of the data stream to accommodate FEC packets and reduce timeouts and the use of FEC on a proactive and reactive basis. We gained insights into the structuring of the proactive and reactive components and developed expressions for the overheads incurred and the expected residual loss rate. We validated the analysis with simulations and found that the analytical predictions closely match the simulation results. We then considered the problem of disruptions and proposed the techniques of *disruption-detection* and *mode-switching* to combat it.

This generic design over an abstract lossy channel was then tailored to the link and transport layers. We designed the link-level protocol based on the analysis and general design. Our design meets the objectives of low link latency, low residual loss rate and high goodput. Such a link introduces minimal interference with TCP mechanisms even under extreme conditions such as disruptions. However, we saw that even with link-level support, end-to-end loss rates are high over multi-hop paths where each link suffers high loss rates. This argued for enhancements at the

transport layer.

The transport layer scheme (LT-TCP) is also structured in a manner similar to the link-level scheme and is based on the analysis. However, due to the need for a reliable protocol and the complexities of TCP, several aspects had to be designed carefully. We looked at appropriate congestion control responses, maintaining TCP semantics, timeout avoidance, packet scheduling among other things. The addition of FEC to TCP provisioned as proactive FEC (PFEC) and reactive FEC (RFEC) enabled us to increase the dynamic range of TCP up to a packet error rate of 50%.

We also looked at an example of disruption in 802.11 networks and saw how capture effects can be detrimental to performance. We studied both cross-system interference with Bluetooth voice calls as well as co-channel interference between two 802.11 WLANs. In both situations, our proposed transport-layer enhancements (LT-TCP) helps improve performance, especially over longer RTTs.

We demonstrated the trade-off between higher persistence at the link-level and reduced goodput at the transport layer (as a consequence of timeouts) and argued for an ARQ limit that provides a good balance between the two. We showed that it is possible to provide a link with significantly reduced delay (under loss and disruptions), high goodput and low residual loss rate and in-order delivery with just 1 ARQ attempt. Over multi-hop paths, we showed how the residual loss rates can accumulate. This causes TCP-SACK, even with our link-layer enhancements, to perform poorly under conditions of disruptions. The support provided by the LT-TCP enhancements (to make transport layer robust to losses) enables us to mitigate this problem. Our results showed the superior performance of the combination of LT-TCP and our proposed link layer enhancements. We also presented the results of performance evaluations under Uniform and Gilbert loss models with focus on key metrics to show the performance of LT-TCP and LL-HARQ compared to baseline protocols (LL-ARQ and TCP-SACK).

We then turned our attention to real world traces. We used real traffic data collected by our research collaborators at MIT-Lincoln Labs to study the impact of our protocols in real-world scenarios. Based on flight data, we modeled the link errors first and gained insight into the behavior of wireless airborne links. We then

developed link models that we plugged into the ns-2 simulator. We could thus test the performance of our protocols under real-world link models. Our results show that LT-TCP and LL-HARQ give significant performance benefits even under such extreme conditions of high and variable losses including disruptions.

Next, we studied the performance of existing link and transport protocols under the impact of noise-induced packet losses on the Open-Access Research Testbed for Next-Generation Wireless Networks (ORBIT) at Rutgers University. This involved using real systems to test the performance of current link and transport protocols. Our results show the severe impact that noise has on the performance of both layers. Moreover, we also provide evidence of the performance degradation in presence of different link loss rates (exported to TCP as residual loss rate), impact of link latency which results in spurious retransmissions at the transport layer.

In summary, we believe we have a set of complementary and robust protocols at the link and transport layers which can provide good performance under extreme environments and lossy conditions including disruptions. As wireless deployments grow and the demand for capacity and quality increases, we believe this work can contribute to realizing the full potential of wireless communications.

10.2 Future Directions

As part of our efforts to provide engineering solutions to performance problems in the link and transport layers, we presented the design of Loss-Tolerant TCP (LT-TCP) to the Internet Congestion Control Research Group which is a working group of the standards body Internet Engineering Task Force. This work was presented at IETF 69 in 2007. Based on the positive feedback from the community, we are currently engaged in efforts to implement the LT-TCP protocol in the BSD and Linux kernels. We hope to finish this shortly and release a reference implementation to the community.

LITERATURE CITED

[1] Atheros. URL: http://www.atheros.com.

[2] Five Keys to Successful Metro-Scale Wi-Fi Deployment. Tropos White Paper,December 2004.

[3] Google Wifi. URL: http://wifi.google.com.

[4] IEEE 802.11s. URL:http://www.ieee802.org/11/Reports/tgs_update.htm.

[5] IEEE 802.16j. URL:http://www.ieee802.org/16/tgs.htm.

[6] MadWifi. URL: http://www.madwifi.org.

[7] Network Simulator 2. URL: http://www.isi.edu/NSNAM/NS/.

[8] Open-Access Research Testbed for Next-Generation Wireless Networks (ORBIT). URL: http://www.orbit-lab.org.

[9] Performance Implications of Link Characteristics (PILC). URL: http://www.isi.edu/pilc/.

[10] Reliable Multicast Transport. URL:http://www.ietf.org/html.charters/rmt-charter.html.

[11] Tcpdump. URL: http://www.tcpdump.org.

[12] The Third Generation Partnership Project (3GPP). URL: http://www.3gpp.org.

[13] Wireless Geographic Logging Engine. URL: http://www.wigle.net.

[14] Wireshark. URL: http://www.wireshark.org.

[15] The CISCO Position on WIMAX and Related Next-generation Radio Technologies for Mobile Operators, 2005.

[16] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level Measurements From an 802.11b Mesh Network. *SIGCOMM Computer. Communications. Review.*, 34(4):121–132, 2004.

[17] M. Allman, H. Balakrishnan, and S. Floyd. RFC 3042 - Enhancing TCP's Loss Recovery Using Limited Transmit, January 2001.

[18] M. Allman, W. Eddy, and S. Ostermann. Estimating Loss Rates with TCP. *SIGMETRICS Performance Evaluation Rev.*, 31(3):12–24, 2003.

[19] M. Allman, S.Dawkins, D. Glover, J. Griner, D. Tran, T. Henderson, J. Heidemann, J. Touch, H. Kruse, S. Ostermann, K. Scott, and J. Semke. Ongoing TCP Research Related to Satellites. IETF RFC 2760, February 2000.

[20] G. Anastasi and L. Lenzini. QoS Provided by the IEEE 802.11 Wireless LAN to Advanced Data Applications: A Simulation Analysis. *Wireless Networks*, 6(2):99–108, 1999.

[21] 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, Leibniz Center, Jerusalem, Israel, June 2004.

[22] A. Bakre and B. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. Proceedings - International Conference on Distributed Computing Systems, page 136, Vancouver, Canada, 1995. IEEE, Piscataway, NJ, USA.

[23] H. Balakrishnan and R. Katz. Explicit Loss Notification and Wireless Web Performance. In *IEEE GLOBECOM Global Interne*, Sydney, Australia, November 1998.

[24] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. H. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. In *SIGCOMM '96: Conference proceedings on Applications,*

Technologies, Architectures, and Protocols for Computer Communications, pages 256–269, New York, NY, USA, 1996. ACM Press.

[25] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. Improving TCP/IP Performance over Wireless Networks. In *MobiCom '95: Proceedings of the 1st Annual International Conference on Mobile Computing and Networking*, pages 2–11, New York, NY, USA, 1995. ACM Press.

[26] C. Barakat and E. Altman. Bandwidth Tradeoff Between TCP and Link-level FEC. *Computer Networks*, 39(2):133–150, June 2002.

[27] C. Barakat and A. A. Fawal. Analysis of Link-level Hybrid FEC/ARQ-SR for Wireless Links and Long-lived TCP Traffic. *Performance Evaluation*, 57(4):453–476, 2004.

[28] D. Barman, I. Matta, E. Altman, and R. ElAzouzi. TCP Optimization through FEC, ARQ and Transmission Power Tradeoffs. In *Proceeding of Conference on Wired/Wireless Internet Communications*, pages 87–98, February 2004.

[29] E. Berlekamp, R. Peile, and S. Pope. The Application of Error Control to Communications. *Communications Magazine, IEEE*, 25(4), April 1987.

[30] A. Bestavros and G. Kim. TCP Boston: A Fragmentation-Tolerant TCP Protocol for ATM Networks. volume 3, pages 1210–1217, Kobe, Japan, April 1997. IEEE.

[31] G. Bianchi, F. Formisano, and D. Giustiniano. 802.11b/g Link Level Measurements for an Outdoor Wireless Campus Network. In *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pages 525–530, Washington, DC, USA, 2006. IEEE Computer Society.

[32] S. Biaz and N. Vaidya. Using End-to-End Statistics to Distinguish Congestion and Corruption Losses: A Negative Result. *Technical Report 97-009, Dept. of Computer Science Texas A&M University*, 1997.

[33] S. Biaz and N. Vaidya. Discriminating Congestion Losses from Wireless Losses Using Inter-Arrival Times at the Receiver. *IEEE Symposium ASSET'99, Richardson, TX, USA*, 1999.

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

[35] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and Evaluation of an Unplanned 802.11b Mesh Network. In *MobiCom '05: Proceedings of the 11th annual International Conference on Mobile Computing and Networking*, pages 31–42, New York, NY, USA, 2005. ACM Press.

[36] S. Biswas and R. Morris. ExOR: Opportunistic Multi-hop Routing for Wireless Networks. In *SIGCOMM '05: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 133–144, New York, NY, USA, 2005. ACM Press.

[37] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. RFC 2475- An Architecture for Differentiated Services, December 1998.

[38] E. Blanton and M. Allman. On making TCP More Robust to Packet Reordering. *SIGCOMM Computer Communication Rev.*, 32(1):20–30, 2002.

[39] 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.

[40] K. Brown and S. Singh. M-TCP: TCP for Mobile Cellular Networks. *SIGCOMM Computer Communications Rev.*, 27(5):19–43, 1997.

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

[42] J. Camp, J. Robinson, C. Steger, and E. Knightly. Measurement Driven Deployment of a Two-tier Urban Mesh Access Network. In *MobiSys 2006: Proceedings of the 4th International Conference on Mobile Systems, Applications and Services*, pages 96–109, New York, NY, USA, 2006. ACM Press.

[43] S. Cen, P. Cosman, and G. Voelker. End-to-end Differentiation of Congestion and Wireless Losses. *IEEE/ACM Transactions on Networking*, 11(5):703–717, 2003.

[44] R. Chakravorty, S. Banerjee, P. Rodriguez, J. Chesterfield, and I. Pratt. Performance Optimizations for Wireless Wide-area Networks: Comparative Study and Experimental Evaluation. In *MobiCom '04: Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, pages 159–173, New York, NY, USA, 2004. ACM Press.

[45] M. Chan and R. Ramjee. TCP/IP Performance over 3G Wireless Links With Rate and Delay Variation. In *MobiCom '02: Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, pages 71–82, New York, NY, USA, 2002. ACM Press.

[46] D. Chase. Code Combining–A Maximum-Likelihood Decoding Approach for Combining an Arbitrary Number of Noisy Packets. *IEEE/ACM Transactions on Networking*, 33(5):385– 393, 1985.

[47] K. Chebrolu, B. Raman, and S. Sen. Long-distance 802.11b Links: Performance Measurements and Experience. In *MOBICOM*, pages 74–85, 2006.

[48] L. Chen, R. Kapoor, M. Sanadidi, and M. Gerla. Enhancing Bluetooth TCP Throughput via Link Layer Packet Adaptation. In *ICC*, 2004.

[49] C. Chiasserini and R. Rao. Performance of IEEE 802.11 WLANs in a Bluetooth Environment. In *Wireless Communications and Networking Conference, 2000. WCNC. 2000 IEEE*, pages 94–99, Chicago, IL, USA, 2000.

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

[51] 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.

[52] R. Draves, J. Padhye, and B. Zill. Comparison of Routing Metrics for Static Multi-hop Wireless Networks. In *SIGCOMM '04: Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 133–144, New York, NY, USA, 2004. ACM Press.

[53] R. Durst, G. Miller, and E. Travis. TCP Extensions for Space Communications. *Wireless Networks*, 3(5):389–403, 1997.

[54] E. Vergetis and E. Pierce and M. Blanco and R. Guérin. Packet-level Diversity - From Theory to Practice: An 802.11-based Experimental Investigation. In *MobiCom '06: Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, pages 62–73, New York, NY, USA, 2006. ACM.

[55] H. Elaarag. Improving TCP Performance over Mobile Networks. *ACM Computing Survey*, 34(3):357–374, 2002.

[56] S. ElRakabawy, A. Klemm, and C. Lindemann. TCP with Adaptive Pacing for Multihop Wireless Networks. In *MobiHoc '05: Proceedings of the 6th ACM International Symposium on Mobile ad hoc Networking and Computing*, pages 288–299, New York, NY, USA, 2005. ACM Press.

[57] G. Fairhurst and L. Wood. RFC 3366 - Advice to Link Designers on Link Automatic Repeat ReQuest (ARQ), August 2002.

[58] K. Fall and S. Floyd. Simulation-based Comparisons of Tahoe, Reno and SACK TCP. *SIGCOMM Computer Communications Review*, 26(3):5–21, 1996.

[59] S. Floyd. RFC 3649 - HighSpeed TCP for Large Congestion Windows, December 2003.

[60] S. Floyd, J. Madhavi, M. Mathis, and M. Podolsky. RFC 2883 - An Extension to the Selective Acknowledgement (SACK) Option for TCP, July 2000.

[61] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla. The Impact of Multihop Wireless Channel on TCP Performance. *IEEE Transactions on Mobile Computing*, 4(2):209–221, 2005.

[62] Z. Fu, X. Meng, and S. Lu. TCP Friendly Rate Adaptation for Multimedia Streaming in Mobile Ad Hoc Networks, 2003.

[63] M. Gast. *802.11 Wireless Networks: The Definitive Guide*. O'Reilly, 1 edition, 2002.

[64] A. Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[65] N. Golmie, R. E. Van Dyck, and A. Soltanian. Interference of Bluetooth and IEEE 802.11: Simulation Modeling and Performance Evaluation. In *MSWIM '01: Proceedings of the 4th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 11–18, New York, NY, USA, 2001. ACM Press.

[66] N. Golmie, R. E. Van Dyck, A. Soltanian, A. Tonnerre, and O. Rebala. Interference Evaluation of Bluetooth and IEEE 802.11b Systems. *Wireless Networks*, 9(3):201–211, 2003.

[67] N. Golmie and F. Mouveaux. Interference in the 2.4 GHz ISM Band: Impact on the Bluetooth Access Control Performance. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 8, pages 2540–2545, Helsinki, Finland, 2001.

[68] A. Gurtov and S. Floyd. Modeling Wireless Links for Transport Protocols. *SIGCOMM Computer Communication Review*, 34(2):85–96, 2004.

[69] W. Hoamouda and P. McLane. An Integrated FEC Coding Scheme for ATM Transmission over Regenerative Satellite Networks. *International Journal of Satellite Communications and Networking*, 23(1):33–46, Jan 2005.

[70] B. Huffake, M. Fomenkov, D. Moore, and K. Claffy. Macroscopic Analyses of the Infrastructure: Measurement and Visualization of Internet Connectivity and Performance. URL:http://www.caida.org/.

[71] Huitema. The Case for Packet Level FEC. In *PfHSN '96: Proceedings of the TC6 WG6.1/6.4 Fifth International Workshop on Protocols for High-Speed Networks V*, pages 109–120, London, UK, UK, 1997. Chapman & Hall, Ltd.

[72] H. Inamura, R. Ludwig, and F. Khafizov. TCP Over Second (2.5G) and Third (3G) Generation Wireless Networks. IETF RFC 3481, February 2003.

[73] S. Iren, P. Amer, and P. Conrad. The Transport Layer: Tutorial and Survey. *ACM Computing Survey*, 31(4):360–404, 1999.

[74] V. Jacobson. RFC 1144 - Compressing TCP/IP Headers for Low-Speed Serial Links, February 1990.

[75] C. Jin, D. Wei, S. Low, J. Bunn, H. Choe, J. Doylle, H. Newman, S. Ravot, S. Singh, F. Paganini, G. Buhrmaster, L. Cottrell, O. Martin, and W. Feng. FAST TCP: From Theory to Experiments. *IEEE Network*, 19(1):4–11, January 2005.

[76] J. Jo and H. Jayant. Performance Evaluation of Multiple IEEE 802.11b WLAN Stations in the Presence of Bluetooth Radio Interference. In *Communications, 2003. ICC '03. IEEE International Conference on*, volume 2, pages 1163–1168, May 2003.

[77] S. Kallel. Analysis of a Type II Hybrid ARQ Scheme With Code combining. *IEEE Transactions on Communications*, 38(8):1133–1137, August 1990.

[78] R. Kapoor, Ling-Jyh Chen, Yeng-Zhong Lee, and M. Gerla. Bluetooth: Carrying Voice Over ACL Links. In *Mobile and Wireless Communications Network, 2002. 4th International Workshop on*, pages 379–383, 2002.

[79] P. Karn, C. Bormann, G. Fairhurst, D. Grossman, R. Ludwig, J. Mahdavi, G. Montenegro, J. Touch, and L. Wood. RFC 3819 - Advice for Internet Subnetwork Designers, July 2004.

[80] D. Katabi, M. Handley, and C. Rohrs. Congestion Control for High Bandwidth-delay Product Networks . In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 89–102, New York, NY, USA, 2002. ACM Press.

[81] T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. *SIGCOMM Computer Communication Review*, 33(2):83–91, 2003.

[82] Kyu-Han Kim and Kang G. Shin. On Accurate Measurement of Link Quality in Multi-hop Wireless Mesh Networks. In *MobiCom '06: Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, pages 38–49, New York, NY, USA, 2006. ACM Press.

[83] T. Kostas. and S. Jordan. Packet erasure FEC on ARQ protocols. In M. Atiquzzaman and M. Hassan, editors, *Proc. SPIE Vol. 4866, p. 126-137, Quality of Service over Next-Generation Internet, Mohammed Atiquzzaman; Mahbub Hassan; Eds.*, volume 4866, pages 126–137, July 2002.

[84] R. Krishnan, J. Sterbenz, W. Eddy, C. Partridge, and M. Allman. Explicit Transport Error Notification (ETEN) for Error-prone Wireless and Satellite Networks. *Computer Networks*, 46(3):343–362, 2004.

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

[86] M. Lacage, M. Manshaei, and T. Turletti. IEEE 802.11 Rate Adaptation: A Practical Approach. In *MSWiM '04: Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 126–134, New York, NY, USA, 2004. ACM Press.

[87] M. Lacage, M.H. Manshaei, and T. Turletti. IEEE 802.11 Rate Adaptation: A Practical Approach. In *Proceedings of the 7th ACM International Workshop on Modeling, Analysis and Simulation of Wireless Systems*, pages 126–134, Venice, Italy, 2004. ACM Press.

[88] P. Lettieri, C. Fragouli, and M. Srivastava. Low Power Error Control for Wireless Links. In *MobiCom '97: Proceedings of the 3rd annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 139–150, New York, NY, USA, 1997. ACM Press.

[89] W. Li, C. Law, and F. Yang. A HARQ Scheme for Combating Burst-errors due to Power Control Gaps in Ka-band LEO Satellite Systems. In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, volume 4, pages 2703–2708, San Antonio, TX, USA, 2001.

[90] S. Lin and D. Costello. *Error Control Coding, Second Edition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[91] B. Liu, D. Goeckel, and D. Towsley. TCP-cognizant Adaptive Forward Error Correction in Wireless Networks. In *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, volume 3, pages 2128–2132, November 2002.

[92] J. Liu and S. Singh. ATCP: TCP for Mobile ad hoc Networks. *IEEE J–SAC*, 19(7):1300–1315, 2001.

[93] M. Luby. LT-Codes. In *Proceedings of the 43rd Annual IEEE Symposium on the Foundations of Computer Science (STOC)*, pages 271–280, November 2002.

[94] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft. RFC 3452 - Forward Error Correction (FEC) Building Block. IETF RFC 3452, December 2002.

[95] R. Ludwig, A. Konrad, and A. Joseph. Optimizing the End-to-end Performance of Reliable Flows over Wireless Links. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE International Conference on*

Mobile Computing and Networking, pages 113–119, New York, NY, USA, 1999. ACM Press.

[96] R. Ludwig and M. Meyer. RFC 3522 -The Eifel Detection Algorithm for TCP , April 2003.

[97] R. Ludwig and B. Rathonyi. Link Layer Enhancements for TCP/IP over GSM. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 415–422, New York, NY, USA, March 1999.

[98] Reiner Ludwig and Randy H. Katz. The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions. *SIGCOMM Computer Communication Review*, 30(1):30–36, 2000.

[99] H. Lundqvist and G. Karlson. TCP with End-End Forward Error Coorection. In *Proceedings of International Zurich Seminar on Communications (IZS)* , February 2004.

[100] H. Lundqvist and G. Karlson. On the Optimization of Local and End-to-End Forward Error Correction. In *Proceedings of Nordic Teletraffic Seminar(NTS)* , August 2005.

[101] T. Salonidis M. Garetto and E. Knightly. Modeling Per-flow Throughput And Capturing Starvation in CSMA Multi-hop Wireless Networks. In *Proceedings of IEEE INFOCOM 2006*, Barcelona, Spain, April 2006.

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

[103] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links. In *MobiCom '01: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 287–297, New York, NY, USA, 2001. ACM Press.

[104] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. RFC 2018 - TCP Selective Acknowledgement Options, October 1996.

[105] David McKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[106] M. Mellia, M. Meo, and C. Casetti. TCP Smart-Framing: using smart segments to enhance the performanceof TCP. In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, volume 3, pages 1708–1712, San Antonio, TX, USA, 2001.

[107] Marco Mellia, Michela Meo, and Claudio Casetti. Tcp smart framing: a segmentation algorithm to reduce tcp latency. *IEEE/ACM Trans. Netw.*, 13(2):316–329, 2005.

[108] A. Miu, J. Apostolopoulos, W. Tan, and M. Trott. Low-Latency Wireless Video Over 802.11 Networks Using Path Diversity. In *IEEE ICME 2003*, Baltimore, MD, July 2003.

[109] A. Miu, H. Balakrishnan, and C. Koksal. Improving Loss Resilience with Multi-Radio Diversity in Wireless Networks. In *11th ACM MOBICOM Conference*, Cologne, Germany, September 2005.

[110] A. Miu, G. Tan, H. Balakrishnan, and J. Apostolopoulos. Divert: Fine-grained Path Selection for Wireless LANs. In *2nd International Conference on Mobile Systems, Applications and Services (Mobisys 2004)*, Boston, MA, June 2004.

[111] G. Montenegro, S. Dwakins, M. Kojo, V. Magret, and N. Vaidya. RFC 2757 -Long Thin Networks , January 2000.

[112] K. Nahm, A. Helmy, and C. Kuo. TCP Over Multihop 802.11 Networks: Issues and Performance Enhancement. In *MobiHoc '05: Proceedings of the 6th ACM International Symposium on Mobile ad hoc Networking and Computing*, pages 277–287, New York, NY, USA, 2005. ACM Press.

[113] A. Nallanathan, W. Feng, and H. Garg. Coexistence of Wireless LANs and Bluetooth Networks in Mutual Interference Environment: An Integrated Analysis. *Computer Communications*, 30(1):192–201, 2006.

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

[115] 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.

[116] K. Park. AFEC: An Adaptive Forward Error-correction Protocol and its Analysis, 1997.

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

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

[119] 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.

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

[121] J. Robinson and E. Knightly. A Performance Study of Deployment Factors in Wireless Mesh Networks. In *Proceedings of IEEE INFOCOM 2007*, Anchorage, AK, May 2007.

[122] V. Roca, Z. Khallouf, and J. Laboure. Design and Evaluation of a Low Density Generator Matrix, 2003.

[123] D. N. Rowitch and L. B. Milstein. On the Performance of Hybrid FEC/ARQ Systems Using Rate Compatible Punctured Turbo (RCPT) Codes. *IEEE Transactions on Communications*, 48(6):948–959, June 2000.

[124] D. N. Rowitch and L. B. Milstein. On the Performance of Hybrid FEC/ARQ Systems Using Rate Compatible Punctured Turbo (RCPT) Codes. *IEEE Transactions on Communications*, 48(6):948–959, June 2000.

[125] Dan Rubenstein, James F. Kurose, and Donald F. Towsley. A Study of Proactive Hybrid FEC/ARQ and Scalable Feedback Techniques for Reliable, Real-time Multicast. *Computer Communications*, 24(5–6):563–574, 2001.

[126] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic Media Access for Multirate ad hoc Networks. In *MobiCom '02: Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, pages 24–35, New York, NY, USA, 2002. ACM.

[127] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. OAR: An Opportunistic auto-rate Media Access Protocol for ad hoc Networks. *Wireless Networks*, 11(1-2):39–53, 2005.

[128] Robert M. Sanders and Alfred C. Weaver. The Xpress Transfer Protocol (XTP): A Tutorial. *SIGCOMM Computer Communications Review*, 20(5):67–80, 1990.

[129] J. Schiller. *Mobile Communications*. Addison-Wesley Professional, 1 edition, 2000.

[130] S. Shellhammer. Packet Error Rate of an IEEE 802.11 WLAN in the Presence of Bluetooth. IEEE P802.11 Working Group Contribution, IEEE 802.15-00/133r0, May 2000.

[131] A. Shokrollahi. Raptor Codes. Technical report, Digital Fountain, Inc, June 2003. Tech. Rep. DF2003-06-001,.

[132] P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. WTCP: A Reliable Transport Protocol for Wireless Wide-area Networks. *Wireless Networks*, 8(2/3):301–316, 2002.

[133] B. Sklar. *Digital Communications, Fundamentals and Applications*. Prentice Hall, 2 edition, 2001.

[134] C. Steger, P. Radosavljevic, and J. P. Frantz. Performance of IEEE 802.11b wireless LAN in an emulated mobile channel. *2003. VTC 2003-Spring. The 57th IEEE Semiannual Vehicular Technology Conference*, 2:1479–1483, April 2003.

[135] R. Stevens. *TCP/IP Illustrated, Volume 1 (The Protocols)* . Addison-Wesley Professional, 1 edition, December 1993.

[136] V. Subramanian, S. Kalyanaraman, and K.K.Ramakrishnan. An end-to-end transport protocol for extreme wireless network environments. In *In Proceedings of MILCOM 06, IEEE Military Communications Conference*, Washington D.C, USA, October 2006.

[137] V. Subramanian, S. Kalyanaraman, and K.K.Ramakrishnan. Hybrid Packet FEC and Retransmission-based Erasure Recovery Mechanisms (HARQ) for Lossy Networks: Analysis and Design. In *Proceedings of WIreless Systems: Advanced Research and Development (WISARD)*, Bangalore, India, January 2007.

[138] V. Subramanian, K.K.Ramakrishnan, and S. Kalyanaraman. Disruption-Tolerant Link-level Mechanisms for Extreme Wireless Network Environments. In *Proceedings of THE SECOND IEEE/Create-Net/ICST International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE)*, Bangalore, India, January 2007.

[139] K. Sundaresan, V. Anantharaman, H. Hsieh, and R. Sivakumar. ATP: a Reliable Transport Protocol for ad-hoc Networks. In *MobiHoc '03: Proceedings of the 4th ACM International Symposium on Mobile ad hoc*

Networking and Computing, pages 64–75, New York, NY, USA, 2003. ACM Press.

[140] H. Takagi and L. Kleinrock. Throughput Analysis for Persistent CSMA Systems. *Communications, IEEE Transactions on [legacy, pre - 1988]*, 33(7):627–638, jul 1985.

[141] D. Tse and P. Viswanath. *Fundamentals of Wireless Communications*. Cambridge University Press, 2005.

[142] Tomoaki Tsugawa, Norihito Fujita, Takayuki Hama, Hideyuki Shimonishi, and Tutomu Murase. TCP-AFEC: An adaptive FEC code control for end-to-end bandwidth guarantee. In *Packet Video 2007*, pages 294–301, November 2007.

[143] N. Vaidya, M. Mehta, C. Perkins, and G. Montenegro. Delayed Duplicate Acknowledgements: A TCP-Unaware Approach to Improve Performance of TCP over Wireless. *Wireless Communications and Mobile Computing*, 2:59–70, December 2001.

[144] Michael Welzl, Mattia Rossi, Andrea Fumagalli, and Marco Tacca. TCP/IP over IEEE 802.11b WLAN: the Challenge of Harnessing Known-Corrupt Data. In *IEEE ICC (International Conference on Communications)*, pages 19–23, Beijing, China, May 2008.

[145] S. B. Wicker. *Error Control Systems for Digital Communication and Storage*. Prentice Hall, 1 edition, 1995.

[146] S. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust Rate Adaptation for 802.11 Wireless Networks. In *MobiCom '06: Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, pages 146–157, New York, NY, USA, 2006. ACM.

[147] G. Xylomenos and G.C Polyzos. Multi-service Link Layer Enhancements for the Wireless Internet. In *Computers and Communication, 2003. (ISCC*

2003). Proceedings. Eighth IEEE International Symposium on, volume 2, pages 1147 – 1152, 2003.

[148] Y. Yang, H. Zhang, and R. Kravets. Channel Quality Based Adaptation of TCP with Loss Discrimination. In *Proc. of the IEEE Global Communications Conference (GLOBECOM '02)*, 2002.